A Context Aware Attack Detection System across Multiple Gateways in Real-Time

By

Joel Scanlan, BComp

A dissertation submitted to the
School of Computing
in partial fulfilment of the requirements for the degree of

Bachelor of Computing with Honours

University of Tasmania

November, 2004

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma in any tertiary institution, and that, to the candidate's knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

…..……..
Joel Scanlan

Abstract

It is understood that intrusion detection systems can make more intelligent decisions if the context of the traffic being observed is known. This thesis examines whether an attack detection system, looking at traffic as it arrives at gateways or firewalls, can make smarter decisions if the context of attack patterns across a class of IP addresses is known. A system that detects and forestalls the continuation of both fast attacks and slow attacks across several IP addresses is described and the development of heuristics both to ban activity from hostile IP addresses and then lift these bans is illustrated. The System not only facilitates detection of methodical multiple gateway attacks, but also acts to defeat the attack before penetration can occur across the gateway range.

Acknowledgments

My supervisor, Jacky, for her continued support and advice over the year, not only with the Thesis itself but also with the two conference papers I wrote.

My co-supervisor Kevin for his assistance during the year with supplying the audit logs for the research and for assistance with technical matters.

My wonderful fiancée Meg who has pushed me onwards over the last few years and has inspired me to apply myself to my uni work.

Both my own parents and Meg's parents for their support over the year through not only a place to lay the head and fill the stomach, but also for proof reading and countless other things.

My fellow honours students who filled the year with countless moments of hilarity.

Virtual World for remaining to be the same great group of guys, willing to take the piss out of each other at every turn.

My Lord and Saviour, Jesus Christ, who is with me always.

Table of Contents

Listing of Figures and Tables

Figures

Tables

1. Introduction

We live in a world which has a digital dimension. The digital dimension is composed of a society which possesses most of the traits of the real-world: people who need protecting, people who protect and people who harm others. In the digital dimension we have countless office workers, students, doctors and people of various occupations who need access to computer systems for their work or play. We also have people whose job it is to provide protection from those who wish to harm computer systems for personal or financial gain.

A Trend Micro (2002) survey of 500 corporations, government agencies, financial institutions, medical institutions, and universities revealed that 85% of them had suffered a security breach in the preceding 12 months. At the height of the MyDoom virus outbreak in January of this year, 1 in 12 emails sent on the internet contained the virus (Leyden 2004). Everyone in the networked world is potentially affected by those who wish to cause harm to computer systems.

However the news is not all bad: there are more people working in the computer security field then ever. Research is constantly underway to invent new and better ways to protect users who do not have the skills to protect themselves. Two systems which are employed to protect users and their networks are Firewalls and Intrusion Detection Systems. Firewalls are designed to keep unwanted traffic out of private networks; while Intrusion Detection Systems are designed to locate, and in some cases remove, the unwanted traffic that may get past a Firewall and access the network.

As networks have become pervasive with computers, through our work and private lives, a large number of networks have reached the point where they actually connect to either internet or other networks in multiple locations. Examples of this are demonstrated by companies or corporations who have networks in multiple cities or countries. Each of these locations where the networks join is called

a Gateway; and when a company has multiples of these locations they are said to have Multiple Gateways. These Gateways are a prime location not only for the companies to run a Firewall, but also a location for other users to attack and try to gain access to the network.

In 2003 Sam Lorimer investigated whether it was possible to analyse the audit log recordings of events across nine connected gateways, to map attacks occurring across the gateways over time. Lorimer (2003) was able to discover that attacks did occur across the multiple gateways, and that it was possible to discover them in real-time. Such Intrusion Detection allows responses to attacks across multiple gateways before they have occurred on each of the gateways in a network of substantial size.

In the vast majority of Intrusion Detection Systems (IDS) currently being developed (or already deployed) attacks against multiple-gateways are not actively being looked for. The current IDS technologies are constantly improving and becoming better equipped at detecting attacks upon a single gateway - not multiple. However, attacks do occur across multiple gateways, and furthermore it is in these situations that the greatest damage can be done to medium and large scale networks. To be able to actively monitor the events occurring at multiple gateways data needs to be correlated from these sources and examined for such attacks; for this analysis to be of any substantial value, it needs to be done in real-time, to allow immediate preventative action to occur. Enabling malicious source IP addresses to be blocked ahead of their attack would greatly lesson their threat to large networks.

1.1 Thesis Aims

The following are the goals which were examined in this study:

1. To validate and confirm the Lorimer System results and detection threshold heuristic through the analysis of larger data sources.

2. To enable the Lorimer system to operate in real-time.
3. To improve the scalability and operational efficiency of the Lorimer System.
4. To create an additional module to act in response to detected attacks against the network.
5. To develop heuristics to efficiently govern the actions that the system takes against malicious source IP addresses.

The Lorimer System enabled detection of attacks against multiple gateways to be detected through the preservation of network context during the centralised analysis processes. As such this context needs to be protected during the changes which are made to the system in the effort to achieve the goals described. These will now be further expanded upon.

Validate and Confirm the Lorimer System results and detection Threshold Heuristic

The Lorimer System results were produced on a relatively small audit log file covering a time period of 16 days (September 2003), and only containing six thousand source IP addresses. As the Threshold Heuristic for determining if a user being likely to probe multiple gateways or not is based on results from this small sample, it is important to re-calculate this heuristical value based on several larger, more extensive audit log files.

Enabling the Lorimer system to being able to operate in real-time

The Lorimer system was built to examine an Audit log from start to finish, extracting all the required data to detect attacks, and then halting activities with the results stored in the database. While this is ideal for a system designed to detect an attack after it has occurred, using an Audit trail, it is not sufficient for a system that has to act in real-time to respond to a current threat to the network.

The first goal of the new system is to become operational in real-time, so that the other goals such as attack response to be achieved, and more specifically attack response. A response to an attack is only truly useful if it can stop the attack from proceeding to cause further damage to the network.

Improve the scalability and operational efficiency of the Lorimer System

The Lorimer system was built to monitor and detect intrusions occurring across multiple gateways by recording selected data to two database tables. As such it has the ability to record every probe which an IP sends against the system; however, such a database grows to be quite large very quickly. The result is not only a large database in terms of storage, but the System itself suffers from the sizeable performance load of having to search and retrieve information from a large database. This performance deficiency results in the system not being able to detect attacks in real-time as it slowly drops behind real-time as the database grows in size.

The creation of an additional module to act in response to detected attacks against the network

The knowledge of the presence of an attack is a highly valuable piece of information; however it is only of real value if it can be acted upon. In a single gateway context, detecting an attack at the gateway and blocking the source IP's access is fairly straight forward, and such defensive measures have been implemented many times (two examples further discussed within this study being PortSentry and Snort (Rowland 2003; SourceFire 2004a)). However, within the context of a multiple gateway attack, a source IP needs to be stopped at gateways it has yet to visit. The actionable information needs to be used in real-time, in the attempt to outpace the speed of the attack, and block a given source IP on the gateways at risk, prior to its attack occurring there.

The development of heuristics to efficiently govern the actions the system takes against malicious source IP addresses

The concept of adding additional rules to firewalls, ahead of an attack upon a series of gateways, when a multiple gateway threat may not result in attacks on each individual gateway, is likely to have some administrators sceptical of the system. The performance cost of the number of rules on firewalls are of great concern to system administrators, and have resulted in rule efficiency applications being produced (Hoffman, Prabhakar & Strooper 2003; Manderson 2004). Thus the idea of adding additional rules to a series of gateways needs to consider the possible cost to performance on those gateways, and the network as a whole. The System will aim to investigate how best to add and remove the rules it creates in real-time, with the length of time which a rule is on a firewall to be as short as possible, while still providing adequate protection. The goal is to maintain the desired level of security through responding to attacks, while also maintaining network performance.

This Thesis will describe the work done to implement the above aims after initially examining the existing literature. The Literature Review placed the proposed work into the context of the current work within this area and discusses approaches that could be used to achieve the goals of this research.

2. Literature Review

The following chapter will examine the existing literature relevant to an Intrusion Detection System across multiple gateways. Topics examined include Audit logs, Single and Multiple Gateway Intrusion Detection, Firewalls and the honour's research carried out by Samuel Lorimer in 2003.

2.1 Audit Log Files

Audit log files have primarily been used in the past to analyse how an attack occurred upon a system after attack has finished (Pfleeger & Pfleeger 2003). Clifford Stoll (1991) in his book *The Cuckoo's Egg* details how he tracked a series of attacks by a hacker on and through his system by printing out his activities and laboriously manually analysing what the hacker had done. During the 1980's audit logs changed from being massive mounds of week long printouts to being stored electronically on the system. Developments in pattern matching techniques allowed for automated analysis of electronically stored audit logs (Anderson 1980); these were the first intrusion detection systems.

During the 1990's advances in computing power enabled the analysis of audit logs to occur in real-time, thus allowing these intrusion detection systems to respond immediately to attacks in some cases (Kemmerer & Vigna 2002). Intrusion detection systems depend on data being logged by the system or network on which they are running to be able to carry out their function. Audit logs are therefore crucial to the defence of computer systems, and, as a result, varying styles and methods of logging have been developed. The two main sources of data collection are *network data* (reading contents of packets on the network, or logging of packets) and *host based* security logs (from operating system, applications or network infrastructure) (Axelsson 2000).

Recording all of the packets sent upon a system for later auditing allows the network administrator to be able to completely re-create attacks in an attempt not only to locate the source of the attack, but also discover the vulnerability exploited (Antonelli, Undy & Honeyman 1998). However, such systems are extremely resource intensive in terms of the storage space required to archive the packets. As a result of these space constraints most software which source data from packets on the network extract the relevant packet information and record it to a host based log file, discarding the packets. The resulting analysis is carried out upon data which is a hybrid of both host-based and network based data; such systems are seen to be the future direction for audit log analysis and Intrusion Detection (Ranum 2001b).

Host based audit logs are produced by several components of the system: operating system, firewall, applications and network monitoring applications. The events which usually result in a log being created in most systems entail identification and authentication mechanisms; creation, deletion or modification of files and directories; network activity and administrative activity relating to processes and account creation (Amoroso 1998).

2.1.1 Log Files

Log files are made in several places, and record numerous events; as a result log files can be quite large depending on the size of the network. The data stored needs to be accessible by administrators or other systems for analysis. To enable this functionality, log files need to be concise. Thus, the information which is stored in audit logs is generally limited to date, time, source IP, destination IP, port, protocol, and result of event. These are required for compliance with TCSEC and the Common Criteria (CSRC 1999).

Log files come in various formats. There is the usual default text file format which is created by most simple firewalls, software applications and other network infrastructure. There is also the Open Database Connectivity (ODBC) format which

allows audit data to be placed directly into a database for analysis. A third format has been produced by several vendors based around the same simple concept with slight variances between the implementations. The idea is that the content of a file is very similar to that of a native text file, however, it requires a special reader program to be viewed or edited thus limiting its accessibility to intruders upon the network. This strengthens the confidentiality and integrity of the audit data stored within (Holden 2003).

Currently the most popular audit logs in use are specific to operating systems. Syslog is a standard UNIX application that not only allows logging by the operating system, but also by other applications running on the system. Windows runs a service called the Event Log. There have been several attempts to create a standardised log format with the following goal: "heterogeneous, transportable across various network protocols, and flexibility sufficient to meet a variety of needs in very different environments must satisfy two basic properties: extensibility and portability" (Bishop 1995, p. 136).

2.1.2 Log Reduction

It is widely known that log files produced by systems do get large. When the time for analysis arrives any decrease in the size of the log is an increase in the speed at which the analysis can occur. There are two main ways in which the log file length can be artificially shortened: Data Filtering and Feature Extraction

Data Filtering

Data filtering is a method by which excess data within a file is removed based upon a set of rules which dictate what data should be kept and what is to be discarded. Static rule sets that allow accurate classifications of audit data are, however, hard to perfect and can not usually be used on another system. In more recent years such rule sets have been produced following research into learning

algorithms and data mining. Such systems have started to have quite a marked success in outperforming "knowledge-engineered" systems (Lee 2002; Lee, Stolfo & Mok 1999). Using data mining to "discover" rule sets does remove the laborious work of manually defining rules on the part of system administrators. However, the question remains as to whether the system is then in a state of waiting for a new method of attack to be developed before collected data would then be extracted for further analysis.

Feature extraction

Feature extraction is a process of log reduction through examining the log file entries, extracting relevant information and discarding the remainder. This method allows the useful data to be extracted from an audit log for analysis, while the excess data is discarded and therefore not used in the analysis process (Denning 1987). Lorimer (2003) used this method when reducing the size of the log files he examined. Lorimer extracted the source IP, destination IP, destination port and the time stamp of events for analysis.

2.2 Intrusion Detection

Intrusion detection is basically the analysis of audit logs, searching for malicious behaviour occurring upon a system or network (Vigna, Giovanni, Valeur & Kemmerer 2003). In the past the analysis was carried out after an attack was complete, in the late 1980's and early 1990's real-time systems were proposed to monitor systems around the clock, to detect attacks as they occurred (Denning 1987). The closer the system is to running in real-time the better enabled it is to detect an attack in progress and notify the administrator or application on the system which will respond to the attack.

The CERT® Coordination Center (2004) reports that over the last decade there has been a 130 fold increase in the number of incidents reported regarding breaches of

security in computer systems. While over the last few decades security in computer systems has got more publicity and users are taking it seriously, attacks upon systems are becoming more frequent. Computers and networked technologies are pervasive through much of the world allowing for attacks to originate from anywhere at any time. The sheer number of attacks results in intrusion detection systems needing constantly to improve to fulfil their role. As a result Intrusion Detection is an active field of research, and will continue to be for the foreseeable future.

2.2.1 Intrusion Detection Accuracy

Intrusion Detection (ID) systems are designed to protect the resources on the networks and computers they operate on. However, as attacks come in a wide variety of forms, using old and new exploits in systems, the accuracy of intrusion detection systems is a concern. When an intrusion detection system makes an error, it is generally doing one of two things: classifying an illegal event as legal and thus not detecting an attack; or classifying a legal event as illegal and thus hindering an authorised user from achieving their desired objective. These two errors are referred to in the literature as *False Positives* (when the user gains illegal access un-detected) and *False Negatives* (when the user is denied authorised rights) (Northcutt et al. 2001; Ranum 2001a). Intrusion Detection Systems, while wanting the best accuracy they can achieve, also have security goals to attain. In some systems, having very high security is an absolute must (the military, for example), and as a result they are content to have a higher probability of a false negative then a false positive. In other systems, where security is more lax, sacrificing security for less probability of a false negative is a preferred option.

2.2.2. Methods of Intrusion Detection

There are two distinct methods by which ID systems analyse audit logs to locate an attack: Anomaly Detection and Signature Detection. Both methods are dependant

upon the audit logs containing the necessary information to analyse events correctly within the system in order to identify attacks. The following section will examine these two methods.

Anomaly detection

Anomaly detection ID systems require a profile of each user or user group to enable the system to "learn" what comprises normal behaviour (Heberlein et al. 1990; Holden 2003). This behaviour modelling or profiling can often be manually configured either by the administrator or by a training period monitoring user activity. Once the system has a users profile they can examine their events within the system logs for behaviour which does not fit the model. For example, an office worker called Joseph may work 9 to 5 every day, always work at his desk and not connecting remotely; he has also never tried to access files on his co-worker Kate's computer. However, one night at 3 am Joseph connects to the office network remotely, and reads all the files on Kate's computer. This is abnormal behaviour, and thus would trigger the Anomaly detection driven IDS, warning that Joseph's account has been compromised.

Anomaly detection is broader then just mapping profiles of human usage. It is also applicable to processes within normal programs upon the system and the system calls they make, allowing them also to recognise programs or processes which don't belong on a given network (Hofmeyr, Forrest & Somayaji 1998).

Anomaly detection models are known to present a greater then acceptable number of false negative classifications, while still detecting specialised attacks (Twycross 2004). In order to counteract this trait the threshold levels for the classification of threats is kept low in many systems to keep the occurrence of false negatives low; however this conversely allows more false positives to occur, decreasing the security of the system.

Signature detection

Signature detection searches audit logs looking for attacks specifically, instead of looking for what is not-normal behaviour, as in Anomaly detection. Signature or misuse detection has a database of attack signatures against which it can compare network event patterns to discover an attack. This results in signature detection systems being able to be operational directly after they are installed without the need for any training of the system (Holden 2003).

Conversely to Anomaly detection, the Signature detection model does not have many false positives, but tends to allow new attack methods through (Twycross 2004). As signature detection requires a database of the numerous signatures, if the database is not kept up to date with signatures and their variants the system is not secure from all attacks.

Signature base intrusion detection is significantly more computationally efficient then anomaly based detection per item of knowledge as it does not need to create matrices for each system activity (Kumar 1995). Brox (2002) comments that signature detection has a flaw in that it requires a signature for a given attack to be able to be detected, and in some instances this is a case of waiting for an attack to occur, to then be able to make a signature to protect against it.

2.3 Multiple Gateway IDS

There are several operational considerations to be dealt with when implementing an Intrusion Detection System which monitors multiple gateways on a single network in real-time. There have been a few systems proposed and researched which allow for communication between IDS which run on the different gateways and then communicate with each other based upon their individual findings (Mounji et al. 1995; Ptacek & Newsham 1998). However these systems, while often covering each host well individually, they do not detect attacks which happen

across the range of gateways as a whole. The ID systems running on each gateway often ignore the significance of a few probes, while in the context of the network as a whole, it is possible that every gateway might be receiving the same minor attack (Lorimer 2003).

The communication that needs to occur between ID systems running on multiple hosts has to be complete. If it is limited only to being what each host "thinks" is important it will always miss events of a wider importance, and does not get an accurate overall view of the systems security (Siraj, Vaughn & Bridges 2004). The method of facilitating the necessary convergence and then central analysis of data from the varied sources is known as *Data Fusion* and *Data Correlation*.

2.3.1 Data Fusion

Data Fusion is the process of collating all of the information from an array of sensors or sources into a single set of data for analysis (Bass 2000). It is upon this combined set of data that analysis can occur in the context of the entire system. Data Fusion techniques can be used in a single host with a Firewall, IDS and packet sniffer as examples of sources; however, in a multiple gateway system Data Fusion enables each gateway (and indeed other computers within the network) to be a source (McCallam, Whitson & Zavidniak 2002).

2.3.2 Data Correlation

Amoroso (1999, p. 146) defines Data Correlation in ID as "the interpretation, combination, and analysis of information from all available sources about a target system activity for the purposes of detection and response." Data Correlation is the process of analysing and interpreting the information collated in Data Fusion. Across multiple gateways this allows for the analysis that occurs to view each attack in the context of the entire system; allowing for the detection of trivial attacks across a network. If such correlation of events is done in real-time it allows for attacks to be caught while in progress and halted.

Multiple Gateway Response

When an attack is detected upon a gateway the result is that action is taken against the source IP, such as blocking the IP address at the Firewall. If this same IP is then detected attacking another gateway on the same network, action needs to be taken to prevent further attacks from occurring on the remaining gateways on the network (Amoroso 1998; Krügel & Toth 2002; Lorimer 2003). The source IP address may have been detected at each gateway, and attacks all stopped before any damage is done. However if an attacker is blocked on 3 gateways on a network, but manages to go undetected on the remaining 4, the security of the network is breached. It is for this reason that IDS's needs to be centralised to maintain the context of the entire network, protecting each contact point with surrounding networks.

Data Transport

An important consideration when transporting sensitive information across a network is always the security of the communication channel being used to transport the data. Within a system where multiple gateways are communicating data describing attacks to a central server, these communications need to be hard to forge, edit or clandestinely removed from the network. If an attacker forged a communication from a gateway, they could then trick the system into believing an attack happened from an alternate source address, perhaps an authorised computer on the network. Similarly if an attacker edited a communication about their own activities, they could then prevent the system from recognising their activities as an attack, and allow their illegal access to continue. An example of a protocol which could be used to provide the needed security to such communications is the Secure Sockets Layer (SSL) protocol (Cheswick, W. R., Bellovin & Rubin 2003).

2.4. Current Intrusion Detection Technologies

Currently in the realm of Intrusion Detection Systems there is a plethora of types and styles for administrators to choose from. There are systems which obviously run signature detection and/or anomaly detection as already discussed; however there are also systems which run inline on the network and monitor network traffic in real-time (Schneier 2000). There are systems which are network based, but operate from audit logs after the events occur; similarly there are host-based systems which operate on data produced by the Firewall or Operating System kernel. Often systems are a mixture of several of these different methods running in a pseudo-real-time such as in the example of Lorimer's (2003) ID analysis.

The following is an examination of 3 current IDS's that are either being developed currently or are already popular in the field.

2.4.1 MINDS

The MINDS (2004) (Minnesota Intrusion Detection System) project has the objective of producing a system which will allow large scale analysis using data mining algorithms to detect attacks (MINDS 2004). The MINDS system uses a combination of signature detection and anomaly detection to provide protection to the University of Minnesota network (UM).

The MINDS system used network traffic flow data collected from CISCO routers. This audit data is then filtered to remove extraneous entries before feature extraction collates the basic required information for analysis (source and destination IP's and ports, protocols, timestamp, flags). Also catalogued is derived contextual information such as the amount of traffic to a destination from a specific source. The extracted, reduced log is then run through the Attack Detection Module of MINDS using signature detection to discover any known attacks. The remaining log is then fed through the Anomaly Detection Modules which allocates a score to

each connection in relation to normal traffic patterns. Connections which score highly are then further analysed by UM network administrators to moderate whether or not the connection was an intrusion or a false positive. Connections which are scored highly by the Anomaly Module, and are not found to be false positives by the administrators, are then further analysed to produce new signatures for emerging attacks. It is in this way the MINDS system is able not only to protect against the more common and well known attacks, but is also very strong on the detection of novel attacks, or attacks which are not yet supported by many other IDS (Ertoz et al. 2003; Ertoz et al. 2004).

2.4.2 NetSTAT

NetSTAT is a Network-based ID system real-time Intrusion Detection System. NetSTAT is designed to be a distributed system to operate in large networks, or networks which contain sub-networks. The NetSTAT model contains the following components: Network Fact Base, State Transition Scenario database, a collection of general purpose Probes, and the centralised Analyser (Vigna, G & Kemmerer 1999). The Probes are located throughout the network, or within the sub-networks of a larger network. They each contain a filter, inference engine and decision engine which have been configured by the analyser. The analyser creates these 3 components for each probe based on the information it gains from the Network Fact Base (knowledge base of the topology of the network) and the state transition scenario database. The location of the Probes is also decided by the Analyser based on the topology as described in the Network Fact Base. Each Probe is customised to its own section, or sub-network, to allow for differing security levels throughout the network.

NetSTAT uses State Transition models to detect when attacks are occurring, or in some cases, whether parts of attacks are occurring. When suspicious behaviour is detected at one probe it can then be flagged and communicated to a neighbouring probe. This communication of a partial attack is aimed at trying to discover the

remaining events to fill the state transition required to match an attack scenario (Vigna, G & Kemmerer 1999). Such communication facilitates detection of attacks across the network while not limiting it to single host detection; however, it does require the probes to find suspicious behaviour in the first instance. While NetSTAT is a step in the right direction for network wide detection, it still requires individual hosts to detect what it believes to be attacks against the entire network.

2.4.3 Snort

Snort is a highly popular platform independent Intrusion Detection system with well in excess of a million downloads (SourceFire 2004a); it in recent years has become one of the benchmarks of single gateway ID systems. Snort contains a Protocol Flow Analyser and a Detection Engine to provide fast, efficient attack detection. The Protocol Flow Analyser allows Snort to know some basic facts about a flow of traffic, such as whether not it is a client or server communication. Once this simple data is gleaned, Snort's then determines whether or not further analysis needs to occur; if it is needed, Snort is then able to be more targeted based on the information gained by the Protocol Flow Analyser. This reduces the amount of inspection required by Snort's Detection Engine, thus reducing the processing required by the system (SourceFire 2004b).

Snort's Detection Engine is a rule-based system for examining packets at the IP and Application Layer using protocol, signature and anomaly-based detection to discover malicious activity. Rules are set to examine packets and their contents searching for specific attack signatures or patterns. Snort is a versatile tool which can be used not only to detect explicit attacks such as buffer overflows, but is also well suited to detecting the more subtle attacks such as port-scanning and fingerprinting. Snort is an open source project which has resulted in it having a highly active community contributing signatures to detect the attacks using the latest vulnerabilities; this could be partially responsible for its popularity.

2.4.4 Agent Based Intrusion Detection

In recent years, several systems have been proposed which attempt to establish collaboration between network-sensors through using agent-based approaches to Intrusion Detection. An example of such a system is Sparta (Security Policy Adaptation Reinforced Through Agents) (Krügel & Toth 2002) which focuses on attempting to provide protection within dynamically changing networks. With various items of computer hardware such as Laptops, becoming increasingly more mobile, it is important to monitor and detect intrusions within networks whose topology is dynamic. Sparta is based around the concept of decentralised analysis across the hosts of the network who each run the mobile agent platform, network sensors and storage for events detected. The process which the agent detects attacks is outlined in Krügel and Toth (2002).

Agent-based approaches have been examined numerous times (Baldi, Gai & Picco 1997; Kahani & Beadle 1997) in the pursuit of allowing network management tasks to be scalable beyond that of smaller networks. Such systems have had a wide range of goals and implementations; however it is a continued area of research and is a direction in which Distributed Firewalls (Section 2.6.1) are moving currently.

2.5 Lorimer's Intrusion Detection

Sam Lorimer's 2003 investigation and implementation examined audit logs from several sources, collated them and then analysed their content for attacks across multiple gateways. He monitored the activity upon the gateways through two modules: the Analysis Module and the Tracking Module.

2.5.1 Analysis Module

The Analysis Module records a real-time state of the number of connections being made to the gateway range. Each time an IP probes one of the network gateways, a

log entry is created in the audit log. As a result of this entry the Analysis Module then creates an entry in the database for the source IP address. If the IP address already exists within the database the Analysis Module increments the count of the number of probes to the network made by that IP address. If the destination IP address is different to the former probe made by the IP, a flag is raised in the database to alert the Tracking Module that a multiple gateway attack may be underway, and that more analysis is required. In a different instance, if the source address probes for a second time and the port number probed is different to the original probe, again a flag is raised in the database to alert the Tracking Module.

The Analysis Module is used to record the total number of times that a source IP address has probed a gateway on the network, and whether or not it has probed multiple gateways or ports. This summary of information could be based upon output from 10 seconds worth of audit log, or a slow week long scan. The timestamp information which is recorded responds to that of the initial probe sent by the user, not the time of any subsequent probes.

2.5.2 Tracking Module

The Tracking Module is used to examine closely a series of entries that relate to specific source IP addresses. These IP addresses are selected based upon the general output from the Analysis Module: source addresses which have probed more then one gateway in the network, or multiple ports on one or many gateways. The Tracking Module provides a much more informative analysis of events by source IP's across gateways, detailing which gateways have been probed, at what time they were probed and on which port. This allows for a more detailed analysis of what source addresses are trying to achieve in their attacks across multiple gateways, and/or ports. For example, Lorimer (2003) was able to detect and monitor a port scan which covered an entire class C address, or 252 destination IP addresses (Figure 2.1, over page) in approximately one minute. Such scans are not detectable using traditional Intrusion Detection systems operating on individual gateways.

2.5.3 Threshold Level

Using the database created by the Analysis Module, Lorimer (2003) was able to calculate an efficient port probe threshold level where after being surpassed, it could be assumed not to be an attack across the gateways, and could therefore be ignored.

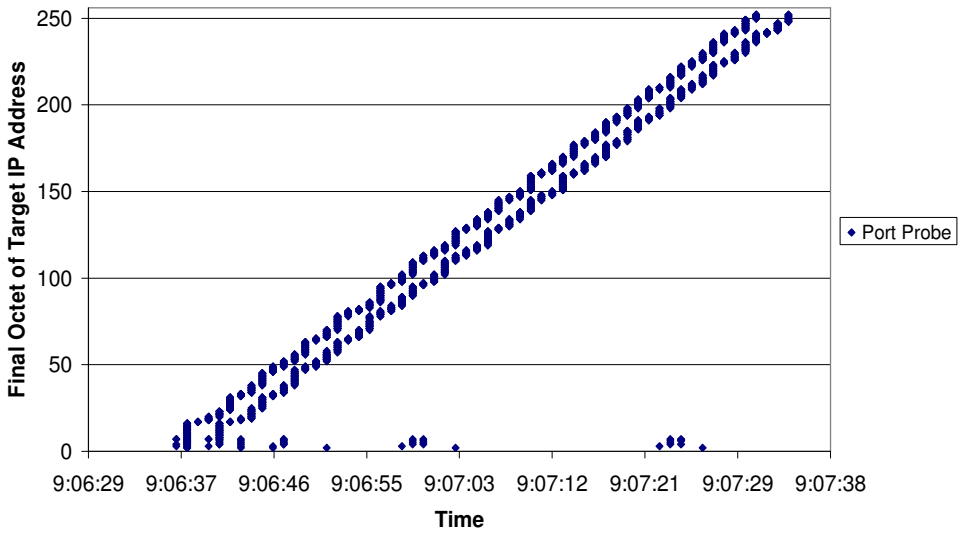


Figure 2.1 Single Source IP Scanning Entire Gateway Range (Lorimer 2003, p. 55)

This threshold facilitates an opportunity for action to be taken to prevent further attacks in the future, without monitoring the activities of an IP indefinitely.

Lorimer's (2003) offline analysis found that a threshold level set at 25 probes provided 100% accuracy on detecting multiple gateway attacks: that is examining only the first 25 probes by an IP, ignoring probes beyond that. Upon examination, Lorimer found that accuracy did not increase in a linear fashion; he was able to conclude that optimum efficiency could be attained with a threshold level of 10 or 11, providing 90% accuracy (Figure 2.2).

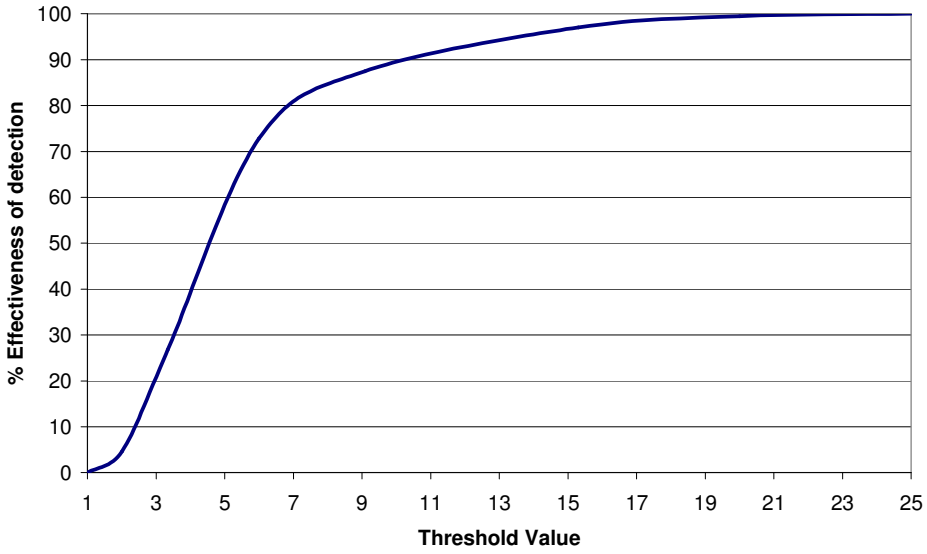


Figure 2.2. Threshold Efficiency Graph (Lorimer 2003, p 52)

2.6 Firewalls

A Firewall, at first glance, appears to be an odd name for a component of computer network infrastructure. The Firewall was originally the name of the physical barrier that is used to separate sections of a home in the advent of a fire, to protect the occupants, and give them time to escape (Avolio 1999). A Firewall in a network also protects its users from danger; it protects users from the danger that lurks in external networks. A network Firewall acts as a filter and blocks traffic attempting to enter into a network, with the goal of only allowing authorised traffic between the two networks.

The first published description of a Firewall was in 1990 by Bill Cheswick, whose goal was to protect the AT&T network from "access to this community by the unscrupulous" (Cheswick, B. 1990, p. 1). The first published reference labelling the style of gateway Cheswick described as a Firewall was in 1992 in the book *Practical UNIX Security* (Garfinkel & Spafford 1991). Firewall developments have continued since their inception as the internet has grown and companies across the globe have expanded their own private networks. The need to keep private and public networks separate is more important now than ever.

A Firewall is the enforcement of a policy; this policy dictates who can enter and leave a network. A Firewall is used not only to protect resources from within the network from attack, but also from export of proprietary information (Ranum 1994). Firewalls are frequently implemented with multiple items of hardware and software and, while simple in objective, are frequently complex in design and implementation. They are usually governed by a list of rules; this list needs to cover a lot of methods of attack from outside. The security of the Firewall depends upon the completeness of the policy implementation.

2.6.1 Types of Firewalls

There are three main methods by which Firewalls operate to fulfil their obligations to their networks: Packet Filter, Circuit Level and Application Level Gateways. These three types of Firewalls are each filtering the incoming traffic based upon a different layer of the OSI model (Al-Tawil & Al-Kaltham 1999).

Packet Filter

Packet filtering Firewalls operate in the Network Layer of the OSI model, or the IP level of TCP/IP. Packets are allowed to continue onto the network based upon the content of the IP header (Chapman & Zwicky 1995). The router or gateway contains a set of rules, and the IP header information is compared to the set of rules. Traffic is then allowed onto the network if it meets the rules, or it is blocked if it does not. In addition to the IP header information (source IP, destination IP, protocol, TCP or UDP source port, TCP or UDP destination port and ICMP message) the firewall also knows the interface the packet arrived upon and the interface it would leave upon.

Packet filters are low cost and low impact on networks (Vicomsoft 2004), however they don't retain any context in the packets they are examining. As a result of this lack of knowledge, the packet filters do not provide the level of granularity that

most networks require to function correctly with only a packet filter based Firewall (Cheswick, W. R., Bellovin & Rubin 2003).

Circuit Level Gateway

Circuit Level Gateway based Firewalls operate at the Session Layer of the OSI model, or the TCP level in the TCP/IP model. Circuit-Level Gateways monitor the TCP/IP handshake within a connection between two hosts and do not actually filter packets itself (Siyan & Hare 1995). Circuit relays are generally used to create specific connections between two specific isolated networks. Circuit-level gateways often work through the internet, allowing for connectivity without detailing information about the private networks route to each other in the un-trusted internet (Cheswick, W. R., Bellovin & Rubin 2003). The way Circuit level operates is that when a user wants to connect to a computer within a given private network the attempt is caught by the Firewall; the Firewall then sets up a connection between itself and the user attempting the connection, and then a second connection between the target and the Firewall. Data is then communicated between the two users through the Firewall, preventing internal network topology from becoming known to the outside user (Stallings 2003).

Application Level Gateway

Application Level Gateway based Firewalls (also called proxies) work at the Application layer of the OSI and TCP/IP models. Unlike the Packet Filter method, the Application layer does examine the contents of packets and can filter based on the actions intended to be taken by individual packets. For example, HTTP packets may be allowed, but the Firewall might filter out all Post commands (Ranum & Avolio 1994).

Configuration of an Application Level Firewall is more complex than the other two alternatives mentioned. Each computer on the network needs to be individually

configured, and each application needs to be setup with the Gateway. Application based systems can tend to be expensive, and also can be less secure if they are not staffed by highly competent personnel (Vicomsoft 2004). Application Level Firewalls tend to be more secure then Packet filter systems as they are only trying to examine the allowed applications upon the Firewall: the packets of other unsupported applications are simply blocked (Stallings 2003). Furthermore, Application Level models allow for ease of logging of network events, thus permitting greater auditing ability of network events.

Other Firewall Types

The above mentioned 3 Firewall types are by no means an exhaustive look at the types of Firewall systems which have been researched or are on the market today. Other systems such as the Stateful Multilayer Inspection Firewall are in fact a combination of the Packet, Application and Circuit Firewall systems. Distributed Firewalls are a field of research leading to each host within a network enforcing its own security policy, which is governed by a central management node (Cheswick, W. R., Bellovin & Rubin 2003). Dynamic Packet Filters are closely related to ordinary Packet Filters, but they address a few of the short falls such as a lack of context of packets within connections.

2.6.2 iptable Rules

A common implementation of a Packet Filter is done through using the rule set known as iptables (Netfilter 2004). The network upon which this study is being completed uses iptable rule based rules as a packet filter.

Iptable rules reside within the kernel and run upon start-up. The rules dictate whether a packet is allowed to enter or exit though a gateway, or whether they are forwarded on to another location. The rules exist in a list and as each packet arrives it is checked against the list, the first match being the action which is used upon the

given packet. Three examples of actions which can be taken upon a packet are ACCEPT, REJECT and DROP.

- ACCEPT – iptables stops searching the list of rules and allows the packet to pass.
- DROP – iptables drops the packet from the network and moves on to the next packet, effectively ignoring the initial packet. (DROP is sometimes referred to as DENY.)
- REJECT – iptables drops the packet and then sends a message to the source IP informing it that it was blocked.

The iptable rules which are generated within this study are expanded upon in Appendix A.

2.7 Attacks

Intrusion Detection Systems and Firewalls are implemented in network infrastructure for one purpose: protection. Attacks upon networks come in various shapes and forms, for various end purposes on behalf of the attacker. Some attacks are extremely explicit in their goal and methodology. A Denial of Service (DoS) attack, for example, frequently presents an option of affecting a system and its owners without needing to actually gain access to the system itself, thus presenting an often easy way to cause damage without needing much experience or knowledge (Scambray, McClure & Kurtz 2001). Other attacks are more targeted and harder to notice due to an attacker wanting to remain undetected by the network owner or administrator. Hackers attempting to steal information or money are an example of this.

Frequently, however, attackers can be detected when trying to gain information for a later attack, probing for exploits and vulnerabilities (Northcutt et al. 2001). Often these actions themselves do not appear overly malicious in action and are ignored;

this is all the more true in a multiple gateway scenario. Two techniques which are used to gather information in a discreet manner are Port Scanning and Fingerprinting. Implementations of these techniques are readily available for download on the internet in the form of applications such as NMAP, Netcat, Strobe, WinScan and SuperScan.

Port Scanning

Port scanning is frequently used to map networks; that is, to find active hosts within a network which are able to be targeted with an attack. After active hosts have been discovered (not all IP's within networks have computers assigned to them) further scans can be completed to discover what ports may be open upon these hosts. The initial scanning to discover hosts can be done using TCP or ICMP ping sweeps. Identifying one these hosts what ports are open requires more selectiveness in the packets being sent. There are several techniques for detecting which ports are open on a host; most of them alter the way in which the TCP 3 way handshake is undertaken. In the TCP 3 way handshake, normally, a packet is sent from the source to the server (SYN), the server then responds (with SYN/ACK), and then finally the user replies (ACK) to finish establishing a connection. An example of how scanning is achieved is through forging a SYN/ACK and sending it to a host, as it never sent a SYN itself, it usually responds with a RST (Reset Connection). Such a packet then allows the attacker to know that the port the SYN packet was sent on is open and listening for connections (Northcutt et al. 2001; Scambray, McClure & Kurtz 2001).

Fingerprinting

Fingerprinting is the technique that allows you to ascertain the operating system which is running on a host through various probes and their responses (Cheswick, W. R., Bellovin & Rubin 2003; Scambray, McClure & Kurtz 2001). Finger printing programs, such as *nmap*, include extensive databases containing the details of the

various differences between TCP/IP stack implementations done by different operating system vendors. Fingerprinting then probes these differences and compares the responses with the knowledge base to produce a probability of what operating system is running on the target host. Once this information is gained, an attack can then be planned against that specific operating system, using its most recent vulnerabilities. In recent times programs have been developed to attempt to fool fingerprinting programs into believing a different operating system is running on the host (Cheswick, W. R., Bellovin & Rubin 2003).

2.7.1. Probing Detection

As a large percentage of the probes detected at gateways are port scanning, fingerprinting and related network mapping techniques, it is important to be able to detect these packets and their intents. In large networks such packets do get into the network past the Firewall; it is at this point that IDS's and other applications are used in their detection. PortSenrty and Snort are used in such situations. Once an attack is detected, a log is created, the packet is dropped and the host is reconfigured to drop all packets intended for the source IP. This response of immediately banning the offending source IP address is labelled a zero-tolerance policy, indicating that the network has no tolerance level for source IP addresses which attack the system.

2.8 Summary

This chapter has examined the wide range of relevant academic literature in the field of Intrusion Detection and Firewalls, with specific attention given to Multiple Gateway Detection. The remainder of this thesis will examine the system upon which this thesis is built, the method for the work carried out and finally the results obtained. All of this is framed within the context of the existing work which was described within this Literature review.

3. Existing Implementation

The work undertaken in this study made use of actual audit data from a gateway range that consists of multiple remote gateways along with the central server (ns1). Ns1 is bound to the IP addresses of an almost complete C-class running from 0 to 252 in the last octet. The resulting IP range of 'virtual' consecutive gateways, as it appears externally to be 253 separate machines when really each IP address will report to the same machine in one amalgamated log (Figure 3.1). The audit log still reports which IP address within this range was probed, meaning that it is possible to analyse the data from this single machine as though it were 253 separate gateways, preserving the context across the network. The majority of the log data recorded in this amalgamated log is simple port probes which have been targeted against IP's within the range. The system that produced the audit logs which this study is examining has a zero-tolerance policy in place (implemented by PortSentry, without put to the log) upon the network (Manderson 2004).

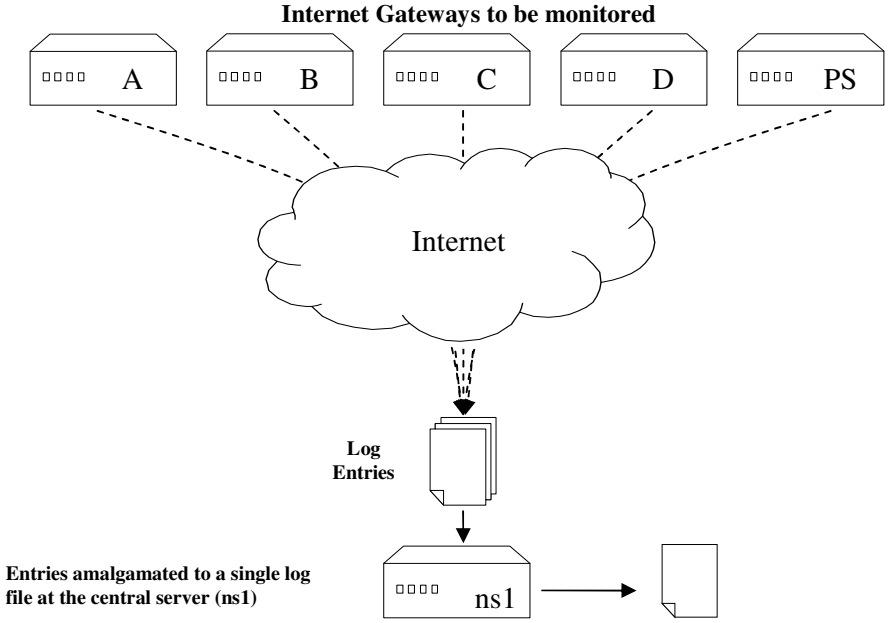


Figure 3.1 The Gateway Log Amalgamation to ns1's central log (modified from Lorimer 2003, p 29)

3.1 Audit Log

As all the log data from each of the gateways across the network feed into a single central machine, it can be examined as if the single machine was the target of all of the attacks against the network thus preserving the network context and allowing for attacks across the wider network to be detected. However, to be able to build a system to parse the log and garner the data to be analysed for an attack, the format of the data within the log file needs to be understood.

3.1.1 Log File Format

The following is an extract (sanitised[1]) from one of the log files used in this study showing the various types of data entries which this study utilized.

Line 1	Aug 1 12:33:49 ns1 portsentry[7688]: attackalert: SYN/Normal scan from host: as132231.bb132.soon-net.com.hk/203.180.13.231 to TCP port: 135
Line 2	Aug 1 12:33:50 ns1 portsentry[7688]: attackalert: Host 203.180.13.231 has been blocked via wrappers with string: "ALL: 203.180.13.231 "
Line 3	Aug 1 12:33:50 ns1 portsentry[7688]: attackalert: SYN/Normal scan from host: as132231.bb132.soon-net.com.hk/203.180.13.231 to TCP port: 139
Line 4	Aug 1 12:33:50 ns1 portsentry[7688]: attackalert: Host: as132231.bb132.soon-net.com.hk/203.180.13.231 is already blocked Ignoring
Line 5	Aug 1 12:33:48 ns1 kernel: Packet log: input DROP eth0 PROTO=1 203.180.13.231:139 203.87.120.198:0 L=84 S=0x00 I=0 F=0x4000 T=49

Figure 3.2 Extract from Central Audit Log

The audit log above illustrates the two log types which we are extracting for analysis: Port Sentry and Firewall. The above log shows the responses from the system to the initial few packets sent by a malicious source IP.

[1] The extract has been edited as it is actual data from an Audit log, and for security reasons it has not been directly reproduced here. The IP addresses have been modified, while the remainder of the log is unchanged.

The first 4 lines of Figure 3.2 are produced by PortSentry in response to actions taken by the source IP address. The first two lines shows PortSentry detecting a port scan on port 135 and then blocking the source IP address (across all ports) as a result. The 3rd and 4th lines show the source IP attempting to conduct a second port scan (this time against 139), however as it has already been banned the scan is ignored as it will be caught at the firewall. The PortSentry entries allow for information to be gathered about an attack in regards to DNS, Source IP, target Port and Protocol but do not allow for the target IP within the range to be known, only that it is within the network's range. This shortfall results in the responsibility of multiple gateway detection to fall primarily on the Firewall log entires.

The last line of Figure 3.2 is an entry which was output from a Firewall upon one of the gateways. As the gateways have a zero-tolerance policy in place; the entries from the Firewall list the action taken against the packet, which in this example is to DROP the packets access. The DROP action is taken by the firewall due to the iptable having a rule to block access to the given IP; such rules are added on this system by PortSentry. The rules are currently generated by PortSentry in response to it detecting an attack through signature detection analysis of packets being sent against the network. The firewall entries contain more information than the PortSentry, listing not only the Source IP address but the Target IP address and the Port which was being targeted. It is for this reason that the Firewall logs are of greater worth for detecting a multiple gateway attack.

As the network gateways are currently operating with a zero-tolerance policy, and it would be unsafe to remove this policy for the study, the source IP addresses being analysed are already banned on the individual gateways of the system. The analysis done in the study will therefore examine the information contained within Firewall audit log entires, ignoring the fact they have been banned locally, searching for the global network context across the IP range.

3.2 Existing System

The Lorimer System, built in 2003, was designed to examine the ns1 Audit log to extract the information regarding network activity across the gateways. Its primary aim was to create a state of the entire network, retaining the context across the gateways, in an effort to detect intrusion attempts at multiple gateway locations. The context was able to be retained through using the raw Firewall and sensor logs. If the Lorimer System was built purely upon output from Intrusion Detection Systems running on each gateway, trivial attacks against these gateways would be overlooked, while at a network wide level they could be of far greater interest. The Lorimer System was composed of two main components: the Analysis Module and the Tracking Module. However, before we examine these twin modules, and how they operated to analyse network activity we will examine how they extract the needed data from the audit log file.

3.2.1 Data Extraction

As Audit log files can grow to be extremely large in size, the system can not merely just view the audit log and analyse it directly; as a result the Lorimer System extracted the required information from the Audit file and entered it into a database in a more succinct format for analysis. This feature extraction was implemented in Perl, making use of its parsing capabilities through regular expressions. As we know the format of the entries within the Audit log, it is possible to extract the entries of interest and discard the rest thus allowing for a vast reduction of the size of the Audit log.

Figure 3.3 (over page) is an example of a regular expression in PERL. This regular expression is used to parse the 4th line of the example log shown in Figure 3.2 (page 28). The expression is segmented to cover the 28 lines to make it comprehensible (with the additional comments) and easily modifiable. The segments which contain

```
($month, $date, $time, $ps_code, $alert, $scan_type, $ip1,
$ip2, $ip3, $ip4, $protocol, $port) =
/          # regexp begins
^          # beginning-of-string anchor
(\S+)      # assigned to $month
[^\d]+     # move to the number representing the date
(\S+)      # assigned to $date
\          # literal space
(\S+)      # assigned to $time
\          #literal space
\S+        # skip over "ns1" text
[^\[]+\[   # move to after the '[' literal
([^\]]+)   # assigned to $ps_code
\]:\       # ']:' and literal space
([^:]+)    # assigned to $alert
:\         # literal string ':' plus literal space
(\S+)      # assigned to $scan_type
[^\/]+\/   # literal string (removes DNS name)
(\d+)      # assigned to $ip1
\.         # literal full stop
(\d+)      # assigned to $ip2
\.         # literal full stop
(\d+)      # assigned to $ip3
\.         # literal full stop
(\d+)      # assigned to $ip4
\ \S+\     # literal string
(\S+)      # assigned to $protocol
\ \S+:\    # literal string
(\d+)      # assigned to $port
/x;        # regexp ends, with x modifier
```

Figure 3.3 A Regular Expression for (used in filtering line 4 of example log)

parentheses are the sections which assign values to the variables listed at the start of the expression. For example the 4 (/d+) statements towards the end of the expression assign each integer octet value to the $ip1-$ip4 variables. Each audit log entry type requires a separate, customised, regular expression to extract the information of interest; the remainder of the regular expressions used within this Study are in Appendix B.

To further illustrate the process of data extraction from the audit log, Figure 3.4 (over page) displays which components of the Firewall log are extracted for storage and analysis. The time and date of the scan are extracted along with the source IP, target IP and target port. The remainder of the log entry is discarded, thus saving a substantial amount of storage space, while the information is then able to be stored in a database allowing it to be more accessible for analysis.

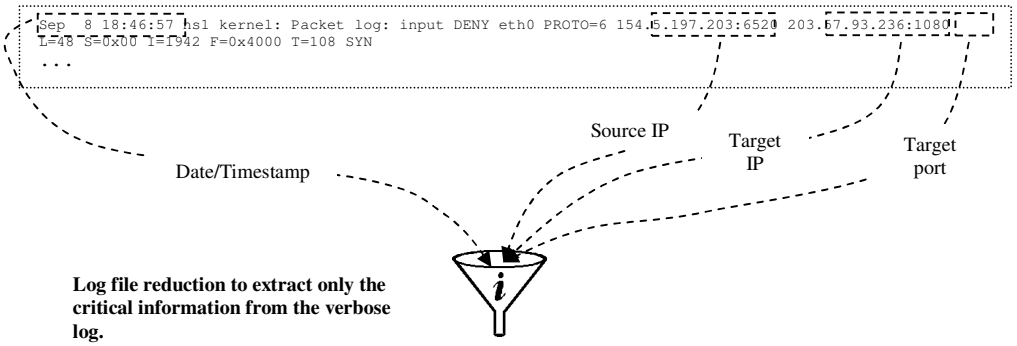


Figure 3.4. Data Extraction from Firewall Log Entry (Lorimer 2003, p 34)

3.2.2 Analysis Module

The Analysis Module is primarily responsible for maintaining the system state of the network at any one time in relation to the number of malicious source IP addresses that have come against the system. In addition to recording the basic information of each source IP address, it also analyses each user's activities to determine whether or not they are conducting a multiple gateway attack.

The Analysis Module works relatively simply, creating a new record for each new source IP address which appears in the Audit log in a database table called the *Audit Table* (Table 3.1, over page). Each time a Source IP reappears in an Audit log entry, indicating a probe or scan, then a count of their activities is incremented. It is the value of this count variable which was used by Lorimer (2003) to establish an effective multiple gateway attack threshold level (see Section 2.5.3 within Literature Review). Each time a new log entry is parsed, the information for target IP and port are compared to the existing values already stored in the Audit Table; if they are different from the original records value then the Boolean values for Gateway Interest and Port Interest are updated. The Analysis Module can then easily and concisely track whether or not a given IP address has probed more then a single gateway or port on the network.

ID	IP1	IP2	IP3	IP4	Date/Time	Target IP	Port	g Interest	p Interest	Count
i	xxx	xxx	xxx	xxx	DD/MM/YYYY hh:mm:ss	xxx.xxx.xxx.xxx	i	boolean	boolean	i

Table 3.1 Audit Module Table

3.2.3 Tracking Module

Once multiple gateway or port attacks suggest to the Analysis Module that a particular source IP address is of interest, then the Tracking Module is used to extract a more comprehensive picture of their activities against the network.

The Tracking Module runs through the entire length of the Audit log file searching for entries which involve a given source IP address. Each entry which involves the given IP is then taken and entered into the database to create a comprehensive record of activity. Each entry into the *Tracking Table* (Table 3.2) contains the Source IP, Target IP, and Target Port thus allowing for a close examination of which gateways or ports the source IP has targeted.

ID	IP1	IP2	IP3	IP4	Date/Time	Source IP	Port
i	xxx	xxx	xxx	xxx	DD/MM/YYYY hh:mm:ss	xxx.xxx.xxx.xxx	i

Table 3.2 Tracking Module Table

3.2.4 Database Tables

The Lorimer System produces two tables of data condensed from the network audit log. The twin tables of Analysis and Tracking cover much of the same information, using the same (or related) column names. The main difference between the two tables is that the Tracking table is far more verbose, storing multiple rows per individual Source IP. Table 3.3 defines what each of the fields within the two tables (as seen in Table 3.1 and 3.2) stores.

ID	A unique sequential identifier given to each Source IP. As port scans can be extremely fast, with the possible of multiple probes within a single second, the combination of Date and Time per IP is not unique enough to act as a primary key.
IP1-IP4	These fields represent the octet segments of an IP address; in the case of the Audit Table it is the IP address of the Source IP, while within the Tracking table it represents the Target IP.
Source IP	As IP1-IP4 represents the Target within the Tracking Table, the Source IP is stored in the Field titled the same.
Date & Time	The date and time of a given network event. In the Audit Table this value is that of the first time a Source IP appeared in the Audit lot; each network event recorded in the Tracking Table has its own time recorded.
Target IP	The Gateway Address which was initially probed by a given Source IP address within the Audit Table.
Target Port	The Port which was probed by a given Source IP address. Within the Audit Table it is the port which was probed in the first entry in the log by the given Source IP Address.
gInterest	A Boolean value stored in the Audit Table representing whether or not a given source IP address has probed more then a single gateway on the network; thus classing it as being 'of interest'.
pInterest	A Boolean value stored in the Audit Table representing whether or not a given source IP address has probed more then a single port on the network; thus classing it as being 'of interest'.
Count	The number of entries within the Audit log for a given Source IP. This is the value which is examined in relation to the threshold level which was proposed by Sam Lorimer (2003).

Table 3.3 Database Field Summary (Bold section for both Tables; Dashed section for Audit Table Only, Grey for Tracking Table only)

3.3 System Overview

The preceding sections have described how the Lorimer System worked technically, Figure 3.5 (over page) aims to illustrate this in a more easily comprehensible manner.

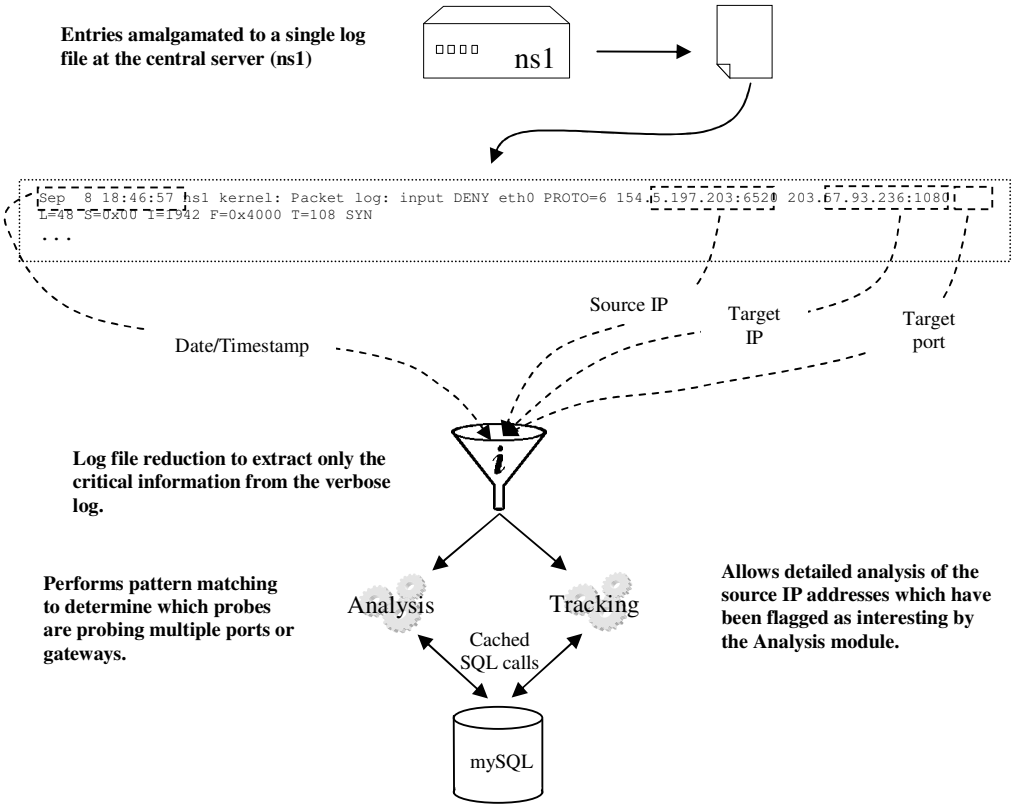


Figure 3.5 Lorimer System Overview (Lorimer 2003, p 40)

3.4 Summary

The Lorimer System, built upon the Audit Logs provides by Kevin Manderson, detects when a source IP attacks multiple gateways upon a single network. Such a system is an ideal platform on which to build a real-time system to act in response to malicious users. The rest of this Thesis will examine the way in which this System was extended to not only operate in real-time, but how to respond to attacks and defend the network in the Methodology. The results of the extensions made to the System are then discussed in the Results and Discussion Chapter.

4. Methodology

At the start of this thesis a series of goals was outlined for this study. The following section will outline the changes and extensions which were made to the Lorimer System to fulfil those goals. The first goal (validating the research done by Sam Lorimer in 2003) required only minor modifications to be done to the system to get it operational; however, extensions to the system were required for the remaining four goals:

- To enable the Lorimer system to operate in real-time.
- To improve the scalability and operational efficiency of the Lorimer System.
- To create an additional module to act in response to detected attacks against the network.
- To develop heuristics to efficiently govern the actions that the system takes against malicious source IP addresses.

4.1 System Modifications

4.1.1 Real-Time Gateway Analysis

The Lorimer System was able to analyse the ns1 Audit log, starting from the beginning of the file, and locate the Source IP address which was a threat to the network in the multiple gateway context. However the Lorimer System did not do this in real-time, and was thus only truly useful at detecting attacks after they had occurred upon a system, thereby establishing an audit trail. To enable the system to respond to an attack as it occurs it needs to operate in real-time; this functionality was implemented into the system using the Perl extension called Tail.

The Tail extension of the standard File module allows a file to be monitored, alerting the Perl program using it to any changes which occur (Grabnar 2004). The extension is perfectly suited to situations where a file's content is being increased,

with additional lines of data being added over time, such as our Audit log. Tail operates in such a way as to attempt to prevent its activities from altering the accessibility of the file to other system processes which may be writing to the file, and are probably the causes for the file to be monitored. Tail measures the amount of time between its most recent data and the previous gain, divides the time by the number of lines added to calculate the average time per line; this average is then used as the time before the next check of the file. However, to prevent Tail from effectively halting its monitoring as a result of calculating a large average due to other system problems, such that data is not appended to the file of interest, a maximum time interval is set to be used if the calculated average exceeds the value. The System has a 5 second maximum time interval.

4.1.2 Changes to Analysis

Tail was primarily installed within the Analysis Module, enabling it to not only detect attacks as they occurred in real time, but also to run for long periods of time unassisted. As the Lorimer System was not run in real-time, it had to be started for each examination of the Audit log. While the functionality of scanning a log from the start has been superseded, it was left within the system to allow for greater ease in testing, or to enable a user to examine past activities prior to real-time analysis if desired. However this is no longer the default and the parameter “start” is required to commence examining the log at the start rather than only monitoring for changes.

As the scalability and efficiency of the system was a concern, and indeed the optimisation of these being a goal, a rather large inefficiency was singled out in the way in which the Tracking Module operated based on the information from the Analysis Module. When the Tracking Module is run, it examines Audit log searching for the Source IP singled out by the Analysis Module. However, if that given Source IP does not appear until half way through the file, or in a worst case, right before the end, it will needlessly search the entire audit log until that point. As a solution, the Analysis Module now stores the byte location of the first appearance

of a Source IP; the Tracking Module can then enter the file at that point and begin building the Source IP's activity profile from there skipping the irrelevant portion of the file.

The Analysis Module's Audit Table also had an additional field added to store the time and date of the last activity by a Source IP address. This is used by both the Action Module and Cleaning Module which will be discussed in Section 4.2.

4.1.3 Changes to Tracking

As already discussed, the Tracking Module now has the byte location of the first entry of the Source IP address of interest passed to it upon the scripts execution; this vastly reduces the run time of the Tracking Module. When the system is running in real-time, without the byte location being passed to the Tracking Module, the system slows dramatically. The system falls behind the required real-time efficiency, with a substantial number of entries being added to the Audit log before the Tracking Module can reach the point at which the Analysis Module requested its analysis of a Source IP's activities, possibly missing the opportunity to catch the attack in progress. Allowing the Tracking Module to (usually) 'leap frog' the vast majority of the log greatly increases the operational efficiency of the Module.

The other addition to the Tracking Modules functionality is that it now stores the time between attacks from a Source IP address in seconds. While this information can be calculated directly in an SQL query, it is more efficient to calculate it once for each entry, than multiple times when searching for attack timing patterns across different Source IP Addresses.

4.2 Additions to Existing System

The main goal of the new system was to produce an action in response to a detected multiple gateway attack, in an attempt to stop further action from

occurring against the network - effectively neutralising the threat. The minor changes to the Lorimer System Modules as outlined above were made to facilitate the implementation of an Action Module which could operate in real-time and was closely linked to the detection capabilities of the Analysis Module.

4.2.1 Action Module

When the Analysis Module detects that a Source IP Address has attacked multiple gateways upon the network it alerts the Action Module which is responsible for taking the required disciplinary measures. The response of the Action Module has to a malicious source IP address is to block its access to each of the gateways upon the network, including those which the IP has yet to attack. As such the Action Module pre-empts the attacker by banning their IP ahead of their attack. The IP addresses to which iptable rules are sent are set by the Friendly Module (See Section 4.2.3).

Ideally, to fully test the system, having a copy of the network from which the audit log comes from would allow for iptable rules to be set on the gateways and test the module's effectiveness perfectly; however due to resource restraints the rules which are created by the Action Module (valid iptable rules) are merely inserted into a database to simulate the number of rules which are in place on the network.

When a Source IP address has been banned, an entry for each rule (1 per Gateway) that has been set is created within the *Banned* database table (Table 4.1, over page). This table lists all of the current bans which are in place across the network for users who have been caught attacking multiple gateways upon the network. The Banned table contains a field which is titled *liftTime* which is used as an expiry date for each rule that is set. If the Action Module continuously created rules without removing bans there would be a substantial hit to network performance as the aged and obsolete rules are continually used to filter traffic through the gateways. The Action Module has a ban length calculation which it computes for each ban it creates; the

method of this calculation is one of the major points of investigation discussed in the Discussion and Results Section.

Once the liftTime has come, a ban has expired, at which point the ban is lifted, removing the rules from the gateways on which it is set; however the information is not at this point discarded. When a ban is lifted, the Source IP address is then added, along with other items of information about it, to the *BanHistory* table (Table 4.2). The BanHistory table stores the past malicious Source IP addresses, which are checked against new attackers appearing in the Analysis Module. If an attack is banned, and is blocked for a time and ceases their attack, the ban against them is lifted. However if they return to the network to attack before their entry has been cleansed from the BanHistory they are re-banned before they can attack multiple gateways. This retains a high level of security across the network for longer without affecting network performance for longer then necessary.

The Banned and BanHistory tables (Table 4.1 & 4.2), similar to the Audit and Tracker tables, share some field types, and as such are described simultaneously in Table 4.3 (overpage).

ID	IP	node	attackTime	liftTime	setTime	lastActivity	activityCount	rule
i	xxx.xxx.xxx.xxx	xxx.xxx.xxx.xxx	0.000	DD/MM/YYYY hh:mm:ss	DD/MM/YYYY hh:mm:ss	DD/MM/YYYY hh:mm:ss	i	string

Table 4.1 Banned Database Table

ID	IP	attackTime	lastActivity	banCount
i	xxx.xxx.xxx.xxx	0.000	DD/MM/YYYY hh:mm:ss	i

Table 4.2 BanHistory Database Table

An additional change was made to the Analysis Module once the Ban Tables were implemented; the Analysis Module now checks each incoming new Source IP Address against the Ban Tables to prevent recording data about an IP which was already banned on the system. Such checking allows for the Cleaning Module (4.2.2) to remove Audit entries once they have been banned across the network.

ID	The ID acts as a Primary key to both tables.
IP	The Source IP address which has been detected attacking multiple gateways and has been banned.
node	The Gateway on which a iptable rule has been put in place, thus the gateway that the given Source IP Address is blocked upon.
attackTime	The attackTime is the average time between each attack launched by a given Source IP Address. It is used as a multiplier to calculate the length of ban to apply to the given Source IP Address.
liftTime	The liftTime is the date and time at which an iptable rule in place on a gateway will be removed, un-banning the given Source IP Address.
setTime	The time at which a ban was put in place by the Action Module against a given Source IP Address.
lastActivity	This details the last day and time that the given Source IP Address was active upon the network. It is used in conjunction with other variables to decide whether or not a given Source IP Address should have its Ban lifted, or remain in place.
activityCount	The activityCount is an integer value which describes how active a user has been since they were banned, incrementing with each activity in the Audit log by the given Source IP Address.
rule	The rule is the exact rule which would be in place on the gateway to block the access to a Source IP. It is a valid iptables rule.
banCount	The banCount is an integer value which describes how many times a given Source IP address has been banned, had the ban lifted, and then been re-banned; a general representation of a repeat offender.

Table 4.3 Ban Tables Field Summary (Bold Line defines a field in both Tables; dashed lines for Banned only and grey for BanHistory only).

4.2.2 Cleaning Module

The scalability and efficiency of the system is vitally important to not only being able to run fast enough to operate in real-time on the current network, but to allow the findings to be transferable to other networks or larger sizes. A key to keeping the system at an acceptable level of operational efficiency is to keep the database as small as possible. To this end, a Cleaning Module was created to work closely with the Action Module removing database entries which were no longer needed.

Once a Source IP is sent to the Action module and banned the Cleaning Module is then responsible for removing the unnecessary Audit Table entries. Likewise it is the Cleaning Module which is responsible for removing the Banned table entry when a Ban is lifted and the Source IP is moved across to be in the BanHistory table.

Over time the Audit table grows with entries belonging to Source IP's which either do not attack multiple gateways or pass the threshold it needs to be cleaned about. To enable the system to judge on which entries a needed to be removes, the Cleaning Module uses a simple heuristic of removing all source IP's after a period of 7 days**.** The motivation for choosing a 7 day period of time to retain source IP activity information has two parts. Firstly, very few attacks monitored during Sam Lorimer's research, or the initial results of this study, showed any attacks which lasted longer then 10 days. It is therefore logical to choose a so a number of days between 5 and 10. Secondly, a large number of ISP's within Australia reset the IP's which are assigned to their subscribers every 7 days, resulting in the users who attack a system changing their IP every 7 days (Manderson 2004).

4.2.3 Friendly Module

As the System is fully automated in the process of detecting and then banning a Source IP from the network, there is always the risk that a 'friendly' Source IP might get banned by mistake. In the situation of an administrator or perhaps even a gateway the results could be quite disastrous, causing a possible self-inflicted denial of service. To prevent such a situation a Friendly Module is made to examine the Audit log and extract all of the Gateway addresses (a few are outside the Class C address range) which were targeted during the attacks against the network. The results from this then form the list used for what gateways rules are sent to when a malicious IP is banned, while also being used as a list of IP's which should never be blocked upon the network.

The FriendlyIP table (Table 4.4) is used to store the list of Friendly IP addresses. In addition to this the table contains a field called *Gateway* which stores a Boolean value which designates whether or not the given Friendly IP is a gateway upon the network. This allows the system to be able to differentiate between which Friendly IPs are gateways upon which rules are to be set, or whether it is just an IP which should never be banned. The FriendyIP table also stores a timestamp for when the IP was added and a count of the number of bans which have added upon the individual gateway. In addition to the automated detection of friendly IPs, an Admin system was built to allow for friendly IP's to be manually added (Appendix C).

ID	IP	timeAdded	gateway	bannedCount
i	xxx.xxx.xxx.xxx	DD/MM/YYYY hh:mm:ss	boolean	i

Table 4.4. FriendlyIP Database Table

4.3 The New System

Below is a visual representation of what the new version of the system looks like in comparison to the Lorimer System shown in Figure 3.5. The Cleaning and Friendly Modules have been combined with the Action Module for simplicity.

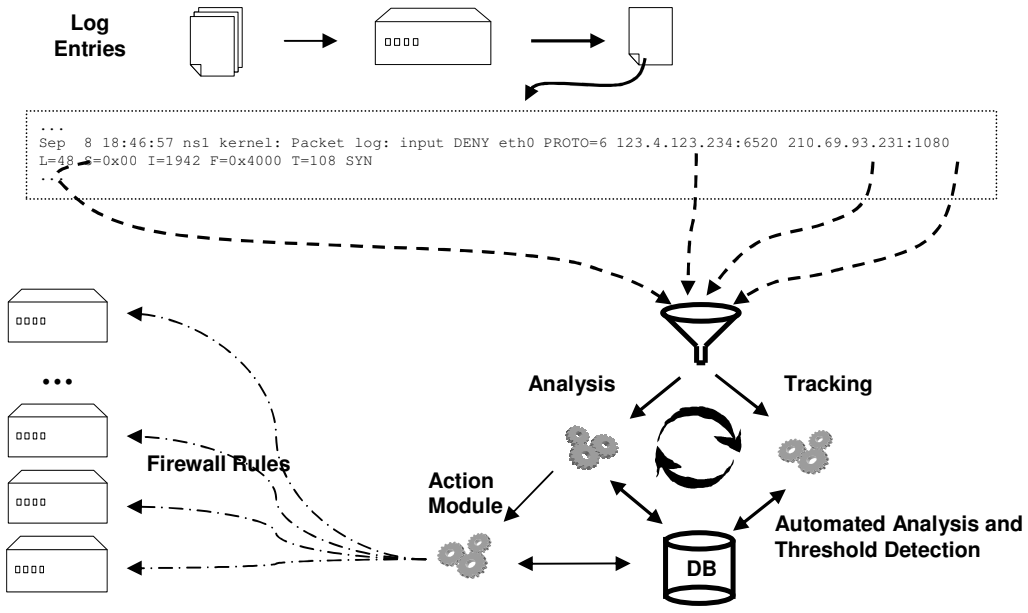


Figure 4.1. New System Overview

4.4 Testing Procedure

The testing that was required to complete the objectives of this study can be broken down into several subgroups. These will be outlined briefly here, and mention given to the way in which they were tested.

4.4.1 Validation

As already discussed, the Lorimer System and its results from 2003 were completed on relatively small audit log file. The first task of this study is to validate the threshold level that Sam Lorimer proposed in his thesis through examining a larger period of time. Table 4.4 outlines the basic statistical size

	Size	Length
Lorimer Log	10MB	10 Days
Log 1	267MB	20 Days
Log 2	187MB	30 Days

Table 4.4 Validation Logs

differences between the 2 log files which are going to be used to validate the Lorimer Threshold, and the log which was used in by Sam Lorimer 2003.

4.4.2 Real-Time Action

The primary area of research within this study was to test the following hypothesis:

"It is possible to detect and respond in real time to attacks upon multiple gateways through the analysis of live data produced by the Firewall and PortSentry using artificial intelligence techniques."

As such the main result gathering and testing carried out in this study is aimed at responding in real-time to threats discovered in a real audit log using the system

developed by Sam Lorimer in 2003. In addition to the Modules already outlined, a Perl script was created to add data entries to the monitored log file from the log file amalgamated by ns1. This effectively made an exact copy of the existing Audit file, one entry at a time.

Once the System was running in real-time the study focus turned to the action that was being taken by the Action Module, and more specifically the length of time a Source IP address was banned. A ban length calculation is carried out by the Action Module for each ban it places; it is the components of this calculation which are going to be varied to discover the optimum time to ban a Source IP Address from the network for attacking multiple gateways.

4.4.3 Efficiency and Performance

The scalability of the system is a sizable concern in being able to act against intruders on large scale multiple gateway networks. The key to making the system scalable to real systems is to make it as efficient as possible in terms of network performance and processing performance on the server. To this end the bans which are set on the network gateways should remain there no longer then needed to preserve network performance, while the database storing the information about Source IP Addresses needs to be kept as small as practically possible without lowering the security levels achieved.

To aid in monitoring the efficiency of the network system an extra table was added to the database for testing the system. The BanLevel table basically maintains the current count of bans in place upon the network and is updated by the Action Module, while also maintaining the history of past levels. This table allows for the progress of the Action Module ban levels to be simply plotted on a graph, allowing for easy comparisons between ban length calculations by the Action Module.

The simplest way to measure the server performance in terms of process and database load is to measure the length of time taken to scan a given series of logs, and the space taken up within the database to store the output from the system. The values from these tests can be compared not only to the Lorimer System, but also with the current system in place on the network and alternate versions of the current system as it was modified during the study.

4.5 Summary

This Chapter has outlined the approach used within this study to examine the goals as outlined in Chapter 1. The remainder of this thesis will examine the results which were attained and discuss their meaning in the context of network security and the goals of this study.

5. Discussion & Results

The following section examines the results which were attained using Audit Log data provided by Kevin Manderson from his Security Consulting Business. The different log files were used with the Intrusion Detection System implemented in Perl which was outlined in the Methodology chapter. The results are in some places compared and contrasted against those which are produced from the Lorimer System which was implemented in 2003, and is outlined in the Literature Review, Existing System and Methodology chapters.

5.1 Validation

In 2003 Sam Lorimer isolated an effective threshold level heuristic by which one could classify whether or not a given malicious source IP was targeting multiple gateways upon a single network. This research was completed using a single Audit log file which covered the 10 day period of the 1st of September 2003 till the 10th of September and was approximately 10MB in size. As this was a limited test case on which to base the findings of his system, the first goal of this research is to validate his threshold level of 11 by using his system on a larger, more extensive audit log file. The file statistics of the logs which are being used throughout this study are listed in Table 5.1 (over page) along with the statistics on the log file which Sam Lorimer used in 2003.

Log 1 and Log 2 are the primary logs for this study, and as such are the logs which are being used to validate Sam Lorimer's threshold level. Log 1 covers a 20 day time period between July 1st and July 21st of 2004; Log 2 covers a 30 day period between August 1st and September 1st 2004. The combined total of the two logs is 50 days of audit log and in excess of 450MB.

	Single Gateway	Multiple Gateway	Size	Length
Lorimer Log			10MB	10 Days
Source IP	5990	776		
% of Total	88.5	11.5		
Log 1			267MB	20 Days
Source IP	67029	8948		
% of Total	88.2	11.8		
Log 2			187MB	30 Days
Source IP	77431	10467		
% of Total	88	12		

Table 5.1 Attacks recorded within the Audit Logs examined in this study

The results showed that not only were multiple gateway attacks also detectable within a larger more comprehensive log, but were actually at equivalent levels to those discovered by Sam Lorimer despite the dramatic difference in file size. There is a minor increase with time; however to validate the increase as being a long term trend a more comprehensive log analysis spanning multiple continuous months would be required. The increase is perhaps likely to be monthly activity fluctuations.

To fully validate the Lorimer results, the optimum threshold level also needs to be calculated on Log 1. During Lorimer's search for a threshold level he found that there was a large drop out of malicious source IP address after 3 probes, similarly Log 1 has a large peak at 3. A threshold level of 3 would result in a detection efficiency of above 90% according to Log 1. The reason for this high detection rate at 3 probes and also why the data shown in Figure 5.1 (over page) appears to start at 3 is related to attackers needing to probe more then a single gateway to be detected, and the vast majority of source IPs probe 2 ports on a gateway before moving on to probe the same ports on the other gateways upon the network. However, as with Lorimer's results discussion in 2003 using a threshold level of 3 would result in ignoring quite a sizeable number of multiple-gateway probing source IP addresses. Figure 5.1 clearly illustrates the efficiency levels for different threshold values within Log 1. The optimum, according to the Log 1 results, would indicate that a threshold around 5 would be the most efficient. This validates the work carried out

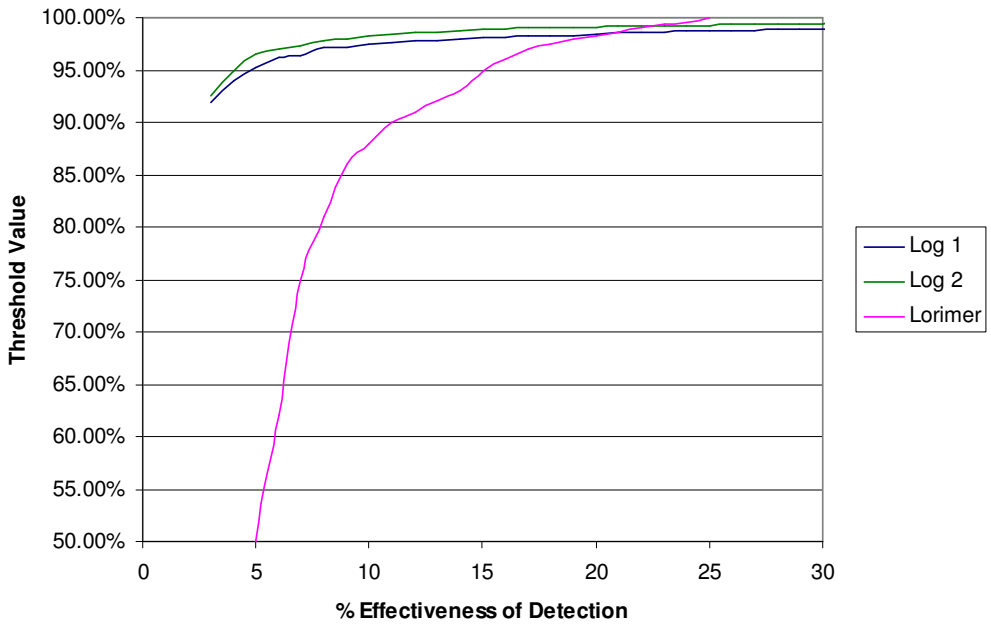


Figure 5.1. Threshold efficiency comparison between the Log 1, Log 2 and the Lorimer Threshold Level.

by Lorimer in 2003, which concluded that a threshold level could be attained which did not require excessive tracking of user activities upon the network. Enabling a threshold level of about 10 or under to detect and then respond to an attack which can frequently last 100's if not 1000's of probes. The Log 2 results (also shown in Figure 5.1 over page) effectively shadows the Log 1 results.

Figure 5.1 also plots the Lorimer Log efficiency line for comparison; it clearly demonstrates the differences between the log files, but also their similarity in relation to threshold efficiency. The lines between the Lorimer Log and Log 1 intersect after the threshold levels defined to be the optimum for both has been passed. Log 1 continues after the Lorimer Log reaches 100 % for two reasons: firstly it is a larger log and would thus contain more instances of users who scan the system for long periods before eventually probing multiple gateways; and secondly, and probably more importantly, the data for the Lorimer efficiency is not complete as we were unable to access the log file he used within his study and he capped his results at 25 probes.

5.2 Action Module

Now that the Lorimer threshold heuristic has been validated, the focus of the study can move to the additions made to the system, of which the item of primary concern is the Action Module. The Action Module, as outlined in the Methodology, is responsible for blocking a Source IP's access to the entire network once it has been detected as being a threat by the Analysis Module. It does this through creating a valid iptable rule and sends it directly to each gateway. Within this study the network and gateways are only simulated, and the rule is merely added to a database; however the rules are valid and could be easily sent through an encrypted SSH session to each gateway in an automated fashion. Resource restrictions dictated that it was not possible to do this during the study.

The Action Module is not only responsible for setting bans but also for lifting bans it has set in place to prevent obsolete rules from affecting network performance. As such the Action module is also responsible for the length of the ban to be set in place for each malicious source IP. The length of ban produced by the Action Module is influenced by two factors: being an long enough to prevent a source IP from causing damage to the system, and short enough to be efficient in terms of the total number of rules in place upon the network so as not to adversely affect the overall network performance. In order to provide the required protection while also keeping a low number of rules in place on the gateways, a dynamic ban length calculation is proposed which will allow each Source IP address to be dealt with individually, on a case by case basis. A static ban length value could well leave a ban upon a gateway far longer then required, or likewise ban a different user for no where near long enough.

5.3 Ban Length Calculation

For the Ban Length Calculation to result in a personalised ban length for each Source IP the factors of the calculation need to be drawn from that IP's activities

upon the network. The Calculation also needs to be scalable to work with the different methods of timing attacks which are present upon the network. Some attacks occur through a fast scan which is completed in a matter of seconds, while other attacks can last hours or even days. These types of attacks need to result in a different Ban Length resulting from the calculation to suit each case equally.

The Tracker table records the time of each packet sent by a malicious source IP, and likewise the Audit table records the time of their first probe along with the total sent. As such it is possible to garner from each table the mean length of time between probes sent by a given source IP. It is this value which will be used as a factor in determining the length of ban to be set in place for a Source IP. In order to gain some idea of what kinds of patterns exist for source IP activity once they are banned, the graph within Figure 5.2 was produced. The graph maps the number of probes sent by a Source IP after it has been blocked from a gateway, showing that the majority of source IP's have ceased their attack after approximately 11 probe attempts. This would suggest that it could be an effective multiplier within a Ban Length Calculation.

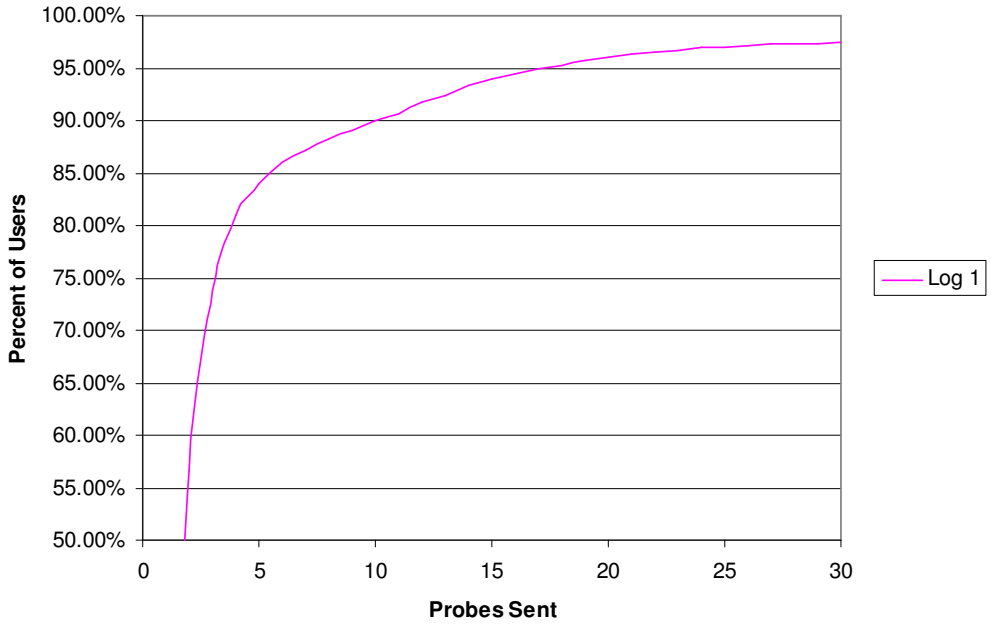


Figure 5.2 Reaction Time for blocked users within Log 1.

A series of calculations, including one based around the value 11, were trialled as possible Ban Length Calculations. They are benchmarked against a 24-hour static value which is being used currently on the network for single gateway IP blocking by PortSentry.

- *Interval Squared:* Squaring the interval allows for a ban length to be calculated from the mean interval while still scaling quite high to provide protection to the network.
- *Interval x Interval / 2:* Similar to Interval Squared, however producing a shorter ban length in an effort to possibly attain maximum efficiency.
- *Interval x Threshold:* While appearing to be chosen for convenience, this calculation is actually based on Figure 5.3 where the approximate optimum point for an interval multiplier is equal to our existent threshold of 11.
- *Static 24 Hour:* This value is being used a pseudo benchmark.

In addition to these calculations, a static value of 100 seconds is added to each result using the mean time interval in the calculation to allow for the cases where a source IP address probes extremely fast and the scan is completed in under a second. This results is a mean time interval of zero seconds when the ban is initially added, thus resulting in the ban being lifted as soon as it is applied.

Figure 5.3 (overpage) illustrates the results from the different ban length calculation methods which were trialled. The y-axis details the number of iptable rules currently in place on the network, while the x-axis details the number of iptable commands which have been sent across the network to add or remove a rule.

The twin squared ban time calculations have resulted in delivering the worst ban length in terms of ban length efficiency. While they are based most tightly on the time interval between attacks, resulting in the biggest difference in ban lengths, they

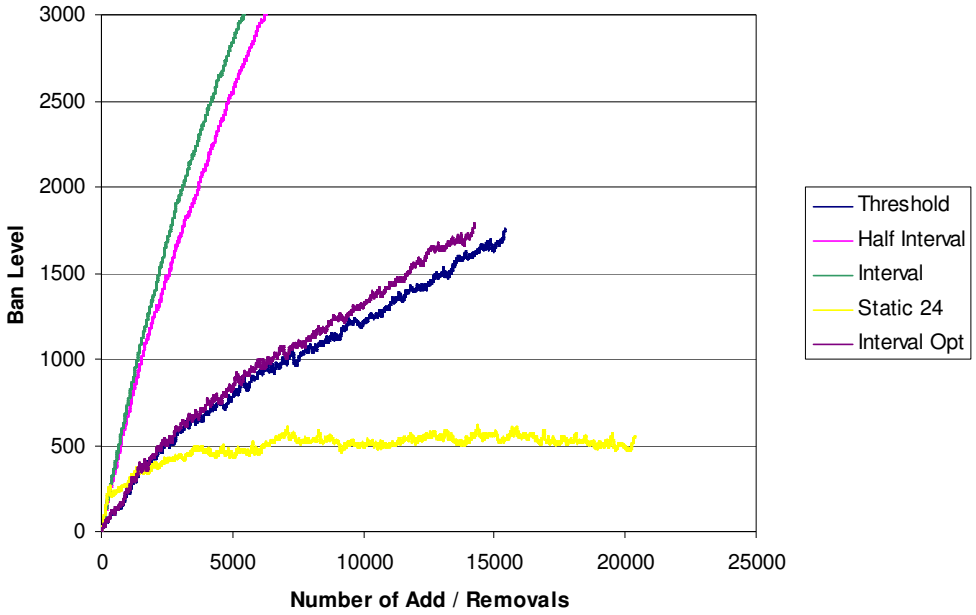


Figure 5.3 Comparison of Ban Length Calculation Methods

very quickly result in bans which are in excess of what is required to stop a source IP's activities. While the ban lengths that are calculated are too long, they are very efficient in terms of the number of rules being sent across the network; however this is of less value due to the extremely small impact each rule transmission has upon the network

The remaining two methods (the static 24 hour ban length and the attack time interval multiplied by the threshold level) each have their strengths and failings.

The static 24 hour ban produced some surprisingly good results; while initially as the worst performer, it quickly became the best in terms of the number of rules upon the network gateways. Now while it is obvious that the static 24 hour ban length results in having a longer ban length than required on short fast scans and that the longer scans require the iptable rules to be re-added multiple times, it still performs well overall. The real strength of the static 24 hour is that it possesses a component the other methods do not have: a maximum ban length. Where the other methods can calculate a ban that lasts weeks – or months in the worst case - it still remains at 24 hours within this method. The result is that if a ban time of a large

length is seen to be needed, but the source IP actually has no interest in a long term attack and ceases their activities, the rule remains on the gateways for a needlessly long time. While these ban times result from the logical scaling up of methods which work in shorter time periods, at larger time periods the methodology which drives the dynamic ban length calculation fails. Despite returning good results in terms of the numbers of rules present on the gateways, it adds and removes rules a lot more frequently then the other methods. The 24 hour ban actually sends 50% more iptable commands then the optimised threshold method discussed shortly.

The Interval multiplied by the threshold method starts off as the best case, but quickly overtakes the 24 hour static ban over the 20 day period. There are two problems one can notice when examining the Interval Threshold method in Figure 5.3: it sends more rule requests to the gateways than necessary and it keeps a lot of rules in the gateways for too long. These two problems are related in that they are the opposite of each other, and a function of the threshold method being too closely constrained by the average case.

The first problem, of sending too many rule requests to the gateways, results from ban lengths which are too short being calculated and which are then lifted before the user has stopped attacking. When a ban is lifted and a source IP attacks again they get re-banned, and then eventually re-unbanned when the rule expires. The result is that 3 iptable commands are sent across the network when only one would have been required with a more accurate ban length calculation.

In an effort to counter this scenario, a version of the system was made which checked for activity by a source IP address in the previous two attack interval periods prior to the expiry time of the ban. If the source IP had been active, the ban length was increased (half threshold multiplied by interval time in this test), and no additional iptable commands were sent. This resulted in over 1200 fewer iptable commands over the 20 day period (as seen in Figure 5.3 as the Optimised line); this was seen to be a significant improvement.

The larger problem with the Threshold ban length calculation is the tendency for ban times of several weeks or months to be calculated. The only method we tested where this was not an issue was the static 24 hour ban, which had the obvious maximum ban time. Therefore a maximum ban length is obviously the solution to this issue.

5.4 Maximum Ban Length

In order to discover the most effective maximum ban length, multiple versions of the Interval Multiplied by Threshold calculation were trialled with differing levels as the maximum ban length. These trials initially examined periods of 3, 5 and 7 days as possible maximum lengths and are shown in Figure 5.4.

The inclusion of a Maximum Ban Length was a very positive result compared to the unbounded ban length methods. The best method without using a maximum length, resulted in excess of 1700 additional rules being in place upon the network,

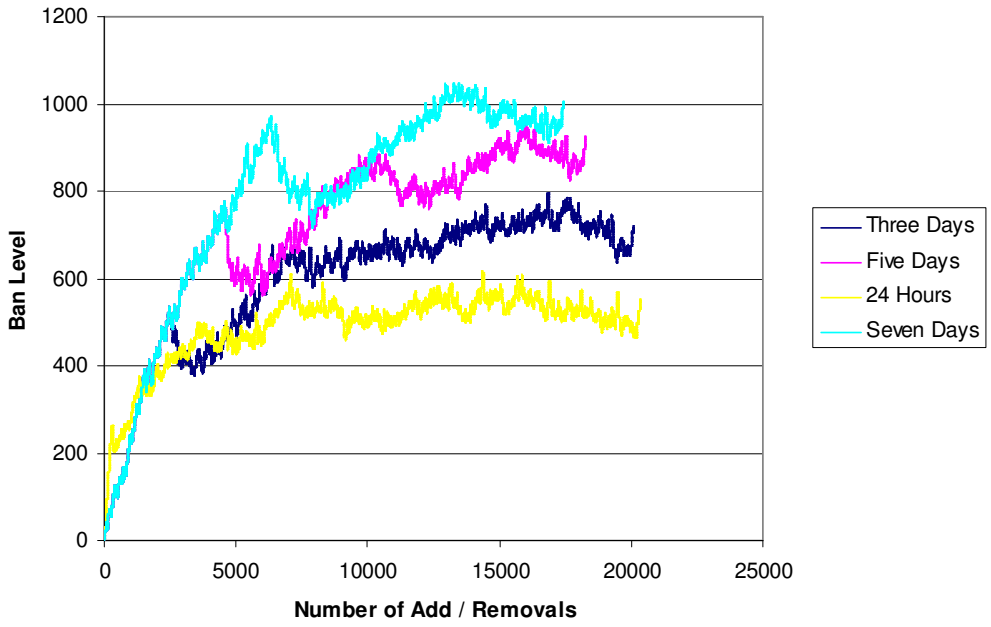


Figure 5.4 Comparison of Maximum Ban Lengths

compared to slightly above 550 on the static 24 hour. However, when using a maximum ban length, the best method returned only additional 700 rules upon the network with a 3 day maximum ban length. While this is not as efficient as the static 24 hour ban length, it is getting close to an acceptable performance. In an attempt to surpass the efficiency of the number of rules in place a 24 hour maximum ban length was then investigated.

Figure 5.5 illustrates the performance achieved using both optimized and un-optimised versions of the 24 hour maximum ban length. The un-optimised version outperforms the static 24 hour ban length in terms of the number of rules in place upon the network. The mean level of bans in place upon the network after 5000 rules had been sent were calculated for the static 24 and the maximum 24 methods. The results showed that the Maximum method averaged 99 less rules in place at 425.7 as the mean, while the static 24 returned 524.7.

Conversely the optimized version of the maximum 24 hour ban returned a ban level in excess of 1300. The optimised version was using the same check that the previous optimised versions had utilized – checking for activity during the previous

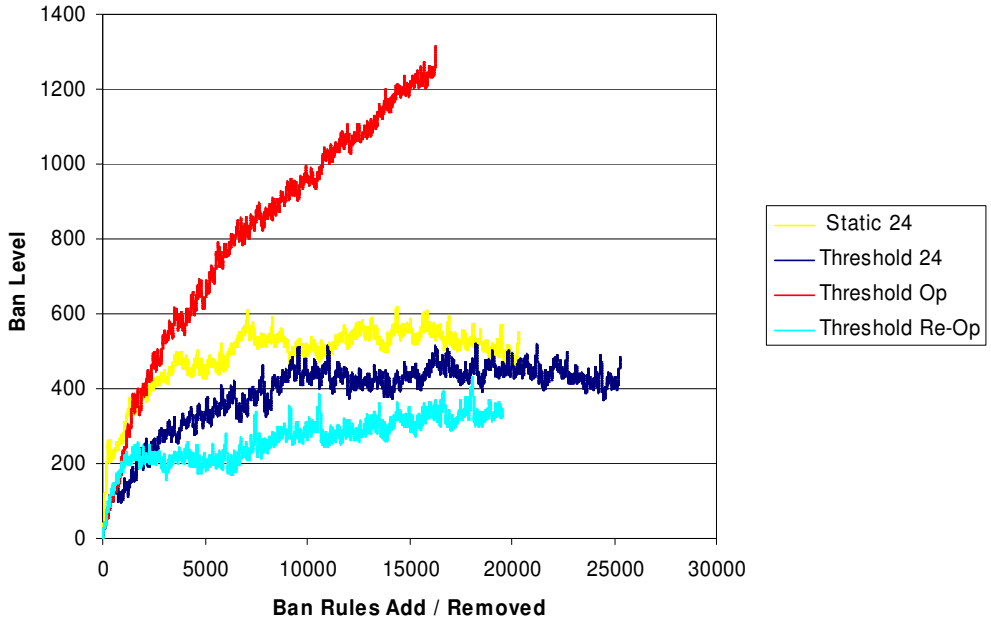


Figure 5.5 Comparison between 24-hour Maximum Calculation and Static 24 hr

2 time intervals for that source IP. However, while it did produce a large saving in terms of rules sent upon the network, the rules in place reached an undesirable level. While an increase in the number of iptable rules being sent in comparison to the static 24 hour length ban is unfortunate, it is acceptable considering the vast time period which the audit log covered; the resulting average period of time between each rule modification being 1 min and 10 seconds.

5.4.1 Optimised Maximum Ban Length

While the Ban Length Calculation has reached an acceptable level, it would obviously be desirable to reduce the number of iptable commands which are sent across the network. It is for this reason that a new optimised version was made with these two key changes to the System:

- Maximum amount of time generated in the lastActivity check is 2 hours,
- attackTime is updated based on information stored in the Banned Table when an IP is moved to BanHistory.

Previously the lastActivity check was often generating lengths of time which resulted in encompassing source IP's who had actually ceased their activities, and unnecessarily increased their ban length. The second change basically recalculates the attackTime based off the length of time the user has been banned, and the number of probes they have sent during that time, and then averages that resulting value with the original attack time. This enables the knowledge which has been acquired by the System during the time a Source IP has been blocked not to be wasted, and can actually go towards calculating a more actuate ban length if they get re-banned at a later date.

The resulting System state which is produced when used with the Log 1 data set can be seen in Figure 5.5 (previous page) labeled as Threshold Re-Op. The results show that not only does the re-optimised version produce a drop in the number of iptable commands sent across the network (800 fewer then the static 24 hour), but also

results in a drop in the number of rules present upon the network. The average number of rules in place upon the network (again averaging the values after 5000 have been sent) is 289.2. This is an additional 130 fewer then the Maximum 24 method which was already 99 in front of the static 24 hour method.

5.5 Scalability

One of the primary concerns in the literature about centralised analysis is its frequent inability to scale. Likewise, one of the goals of this study was to improve the scalability of the Lorimer System, not only to allow it to work on larger networks, but also allow it to work effectively in real-time. The following section will outline the results from a few changes made to the system and the addition of the Cleaning Module. The tests were carried out for the most part on a 50MB file which was an extract of the first few days of the 20 day Log 1 Audit file. The scans were done in a faster then real-time simulation allowing the marginally under 5 days of log to be examined in an average time of about an hour.

5.5.1 Tracking Module

As outlined in the methodology the Audit Table now includes an extra field to optimize the Tracking Performance. Formerly when the Analysis Module discovered that a given source IP was of interest, it would alert the Tracking Module which would then proceed to search the entire Audit Log File for entries relating to the given Source IP. However, now the Audit Table stores the location of the first entry within the log of each IP. The Tracking Module can then reference this value and search forward through the file frequently skipping the vast majority of the file producing a far more acceptable runtime. Table 5.2 contains a comparison between the runtime of the Lorimer System and the resulting system with the byte location optimisation for the Tracking Module. The performance gain is a 36.7% faster system scan, returning the results 54 min faster then the Lorimer System.

	Time Taken	Database Size	Tables in use
50MB / 4 Day			
Lorimer	2:26:06	35396 rows	Audit, Tracker
Lorimer Optimised	1:32:32	35396 rows	Audit, Tracker
Lorimer + Action	0:43:51	21387 rows	Audit, Banned, BanHistory
Action + Cleaning	0:45:13	20650 rows	Audit, Banned, BanHistory
Log 1 (267MB / 20 Day)			
Lorimer + Action	12:39:51	84262	Audit, Banned, BanHistory
Action + Cleaning	11:23:58	43329	Audit, Banned, BanHistory

Table 5.2 Runtime Performance and Database Size of System Variations

As the focus of this study was upon taking action against users who were a threat the network as a whole, and not discovering new patterns or developing individual source IP profiles the remainder of the tests were done without the Tracking Module being in use. The Action Module does not require the Tracking Table in the process formulating the rules protect the network. In a comparison between the optimised Lorimer system and the System once it had the Action module (and its related tables) added, the system performed substantially faster completing the test in less then half the time. The time saving of in excess of 48 min was not the only gain with a 40% reduction in database storage required. Figure 5.6 illustrates the progression in system performance across the differing implementations described in Table 5.2

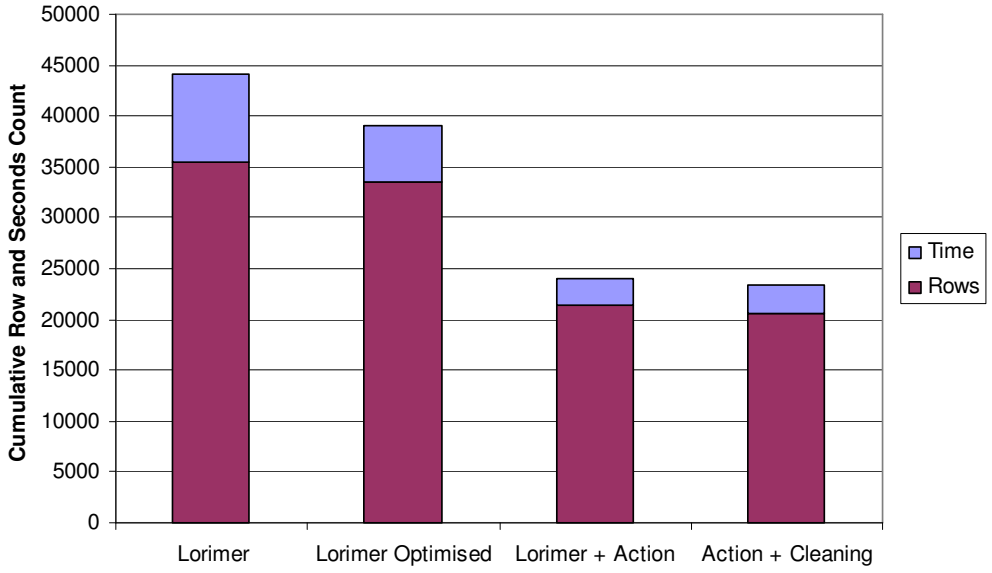


Figure 5.6 System Performance Comparisons across the various implementations

5.5.2 Cleaning Module

In addition to the modifications made to the Tracking Module, an additional module was added to the system which was responsible for cleaning the database of obsolete and unnecessary entries. Table 5.2 outlines the results on the size of the database after the usage of the cleaning module. Under the 50 MB log a difference between the Cleaning and Non-Cleaning versions of the system are negligible, with only a 700 row saving and operating 1 min and 20 seconds slower. The reason for this very minor difference is an illustration of the constraints of a small dataset.

The 50MB log does not cover a great enough period to allow for the removal of Audit Table entries which have expired by passing our 7 day limit; thus to test this the full 20 day log was trialed. The results listed in Figure 5.6 (and Table 5.2) show not only that the Cleaning module reduced the database size by 49%, but also completed the simulation over an hour faster. The speed increase is related to the fact the Audit table is much smaller, thus returning results from queries faster. However, as the Cleaning Module is removing knowledge from the system it is also important to consider the performance loss in terms of classification of malicious

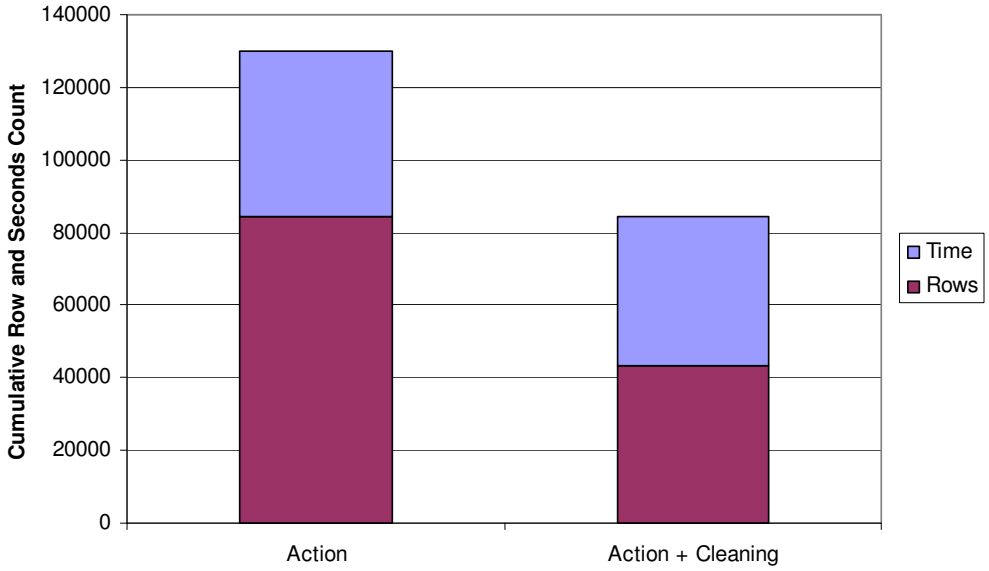


Figure 5.7 Improved Performance Yielded over 20 days through the usage of the Cleaning Module.

source IP addresses. Upon examining the resulting Banned and BanHistory database tables there was a 5% fall in IP's blocked from the network. While this fall in accuracy by itself is not totally unacceptable, it is worth noting the possibility of the value having an added uncertainty. Part of the justification of choosing a 7 day expiry rate for source IP's is that a large portion of Australian ISP's (and indeed others around the world) change the IP of broadband users every 7 days, with dialup users frequently having limited sessions with a new IP each session. As such, it is possible that two different users probe the system from the same IP, each probing a different gateway, which would then appear as a multiple gateway to our system. Such a situation, while unlikely, is a possible scenario which maybe a component within the 5% error rate of the Cleaning Module.

5.6 Other Findings

5.6.1 REJECT vs DROP

After examining the way in which malicious source IP's reacted to being banned (Section 5.3) from a gateway, and how long they continued to attack before ceasing their activities, it was decided to examine what their response would be to the iptable rule being set as a REJECT, instead of a DROP in the context of a multiple gateway attack. Log 3 which is outlined above showed a vast increase in the number of malicious source IP addresses which targeted multiple gateways. The REJECT sends a response to the source IP saying it is not allowed at the given gateway, whereas the DROP merely drops the packet without alerting the user. As expected the source IP's continued their attack against the individual gateways; being assured of the gateway's presence. However due to the analysis across multiple gateways it was possible to show that that the source IP's attacking multiple gateways had a 350% increase in volume. This would indicate that more malicious source IP addresses are interested in executing a network wide scan then the 11-12% values as shown in Table 5.1 (Section 5.1) for the Lorimer Log, Log 1 and Log 2.

	Single Gateway	Multiple Gateway	Size	Length
Log 1			267MB	20 Days
Source IP	67029	8948		
% of Total	88.2	11.8		
Log 3			68MB	5 Days
Source IP	19740	14240		
% of Total	58.1	41.9		

Table 5.2. Log 3 and Log 2 file statistics contrasted.

As the Log 3 established that the REJECT rules encouraged malicious source IP addresses to continue their attack against the system. Figure 5.6 was made to establish visually the difference in the length of time taken for these source IP addresses to cease activities upon the network.

5.6.2 Attacking Timing

While the Tracking Module is not being used by the System itself to take Action against multiple gateway attackers, it is not of no use. The Tracking Module records a very verbose record of what each Source IP address has actually done, recording the gateway probed, port and the time it occurred. Through manual examination of this record it is possible to detect patterns which exist in the methods of users who have attacked the network.

An example of the patterns that can be garnered from the Tracking Modules record is in the timing of when attacks take place. Table 5.3 (over page) lists the time intervals (in seconds) between probes which were carried out by 4 different source IP addresses. Each attack shared the following traits:

- each lasted 18 probes,
- each probed 2 network gateways,
- each can be broken down into 9 episodes of 2 probes each, and
- each had very similar timings for the attacks.

ll 4 attacks can be broken into 9 episodes where 2 probes are sent very quickly, followed by a far larger amount of time with no activity until the next episode takes

place. The first episode in each case is followed with a very large time period of 85760 seconds (approximately 23 hours, 48 min), after which the second episode begins. The first episode in each attack targets a different gateway to the remainder of the episodes, thus having the first 2 probes sent to one gateway and the remaining 16 probes sent to a different gateway. The remaining 8 episodes are separated by a time period of approximately 21600 seconds (6 hours). In several of the instances the longer time periods are exactly the same across different source IP addresses. The time interval between the 10th and 11th probe (recorded in row 11) has a standard deviation of 2.6 seconds or 0.01% of the total time interval for the period between probes.

#	IP1	IP2	IP3	IP4	Average	STD	%
1	---	---	---	---			
2	3	3	3	3			
3	85757	85757	85789	85742	85761	19.81	0.02%
4	3	3	3	3			
5	21527	21519	21484	21536	21517	22.75	0.11%
6	3	3	4	4			
7	21637	21641	21682	21621	21645	25.98	0.12%
8	3	3	3	3			
9	21564	21587	21521	21579	21563	29.42	0.14%
10	3	3	3	4			
11	21587	21592	21586	21588	21588	2.63	0.01%
12	3	4	3	3			
13	21619	21618	21637	21616	21623	9.75	0.05%
14	4	3	3	3			
15	21569	21594	21596	21592	21588	12.61	0.06%
16	3	3	3	3			
17	21591	21593	21580	21596	21590	6.98	0.03%
18	3	4	3	4			

Table 5.3 Differences in the Timing of Attacks across four source IP Addresses.

The visible differences between the probes are also not necessarily present by choice by the attacker. For example, the second probe in each episode is listed above to occur 3 seconds after the first and at other times 4 seconds. This value is probably the same length of time in each instance as it is rounded to the nearest second by the Audit Log. Secondly, the larger time differences would be affected by what other processes would be running on the source machine which may impact on it

measuring the time between probes. Likewise network latency at the time of a probe being sent could slow the time taken for it to arrive at its destination.

The correlation between these 4 attacks, from source IP addresses within the same Class-C address is quite remarkable. However, while it is not a standard case, it is also not a freak, "one off", occurrence. A similar pattern was first recognized within a 10MB log involving three source IP addresses and motivated the examination of the 50 MB file for similar patterns. Likewise it was not the only such example within the 50 MB file.

It is well within the realms of possibility that similar patterns could be established through the examination of the order in which gateways are probed, or a combination of the order of gateway probing and the ports being probed in conjunction with timing to develop source IP profiles to be able to recognise and detect users which change source IP addresses between attacks. Blocking a single IP does not thwart their attacks, or limit their knowledge acquisition about a given network, as their information is centralised behind multiple source IP addresses.

6. Conclusions and Further Work

The research within this study continued the work started by Samuel Lorimer in 2003 in the detection of attacks against multiple gateways through the analysis of amalgamated Audit log files. The system which was produced expanded upon the Lorimer Detection system to not only run in real-time, but to also respond to malicious source IP addresses who are a threat to the system. The study examined 3 main areas: Validation, Action and Scalability; these results will be summarised.

The initial goal of the study was to validate the results found by Lorimer in 2003 whose main discovery was that of an effective threshold level to use in the detection of malicious source IP addresses targeting multiple gateways upon a network. The proposed threshold of 11 probes was validated with 2 Audit log files each far larger then the original Lorimer study log. Both of the test audit logs returned results indicating that the 11 probe threshold level was an efficient and acuate detection threshold level to be used.

Once the threshold level was validated the work then turned to taking action against those who were detected as a threat against the system as facilitated by the threshold level within the Analysis Module. The response against the malicious source IP addresses was to block their access, not only to the gateways which they had already attacked, but upon all the gateways on the network in an effort to pre-empt any future attack. The Action Module was built to provide this protection, which it achieved through the generation of iptable firewall rules to be set on each gateway.

While the system can be protected easily by setting iptable rules, network performance can be dramatically affected if the number of rules increases dramatically. The number of rules in place on the network needs to be kept to a minimum, with obsolete rules being removed when they are no longer valid. To enable this functionality to occur this study examined the usage of an expiry date

for the rules the Action Module sets upon the network. Various methods were examined with the results obtained from a calculation which multiplied the average time between probes sent by the given source IP and the 11 threshold level (average number of probes sent after an IP has been blocked) with a maximum time of 24 hours. Initially the trialled methods did not include a maximum ban length and resulted in bans being formulated which lasted weeks and even months in a number of cases. The result of such methods was having 3 times the number of bans in place on the network then the current static method employed on the single gateways of the network used in the study. The final chosen method (an optimised version of the threshold multiplied by mean interval with a maximum of 24 hours) provided a performance increase over the method currently in place upon the network while still providing the additional security.

During the process of examining the way in which malicious IP addresses reacted to being blocked from the network a trial was done with changing the iptable rules from being a DROP command to that of a REJECT. Not only did this increase the number of probes sent by a blocked IP against a given gateway, as expected, it also vastly increased the number of malicious source IP addresses interested in multiple gateways as a whole. The DROP rule results in 11% of source IP addresses scanning multiple gateways upon the network, while in excess of 40% attacked multiple gateways under a REJECT firewall rule.

The third main area of investigation within this thesis was an examination of the system in the context of its scalability, and how improvements made to the system increased its scalability. The system produced by Sam Lorimer was run through on a 50 MB log file measuring the time and database storage space used by the system. These numerical values were recorded and then compared with the same results from trials run upon optimised versions and versions of the system with additional modules. The results showed a vast improvement in the amount of time taken and storage space requirements for the system. As such the final system produced is not

only more scalable to larger network environments then the Lorimer system, but is also better equipped to operate in real-time.

In conclusion the study was able to show that it is possible to react in real-time to threats against multiple gateways through the use of an intelligent threshold heuristic, while also responding in such a manner as to preserve network performance through an optimum ban time calculation. Furthermore it was possible to complete these additional actions with a reduced computational impact upon the centralised analysis processes through optimising the storage requirements.

6.1 Further Work

As it has now been shown that not only can source IP addresses can be detected but banned in an efficient manner using a simulated environment, based upon real audit logs, the next step is to run the system live upon a network with multiple gateways. Such further work would not only further validate Sam Lorimer's detection threshold, but would also validate this entire thesis in terms of the ability to respond to an attack in real-time. Such testing would also allow for network performance to be monitored with the system running, adding rules upon the gateways, in an automated fashion.

However, the further work is not merely limited to attempting to run the system in a practical situation, but numerous other research directions which could be investigated. The most prominent of these would be the continued examination of the methods used in banning users who attack a network, and the length of time for which they are banned, or that their History is stored by the System.

Attack Timing (Section 5.6.2) described the presence of attack patterns which could be extracted from the Tracking Table produced from the Audit log. If such analysis could be done in an automated fashion, to the end of occurring in real-time, the protective measures which the system could take would be vastly improved against

such organized malicious users. While such attackers would probably only make up a small percentage of the total users attacking the network, they are possibly the greatest threat as their methods are aimed at avoiding correlated detection of their activities.

A future direction which the system could benefit from greatly would be the development of agents to run upon the gateways to extract the audit data and transmit it to the central processing server. This would save amalgamating the logs and then scanning through them with the Analysis Module extracting the required data. If the required information was sent without the excess information it would reduce the work load required by the centralized Analysis Module. This would further to increase the systems scalability and efficiency to allow it to be implemented on even larger networks without reducing the overall effectiveness of the systems detection mechanisms.

7. References

Al-Tawil, K & Al-Kaltham, IA 1999, 'Evaluation and testing of internet firewalls', *International Journal of Network Management*, vol. 9, no. 3, pp. 135-49.

Amoroso, EG 1998, *Intrusion detection : an introduction to Internet surveillance, correlation, traps, trace back, and response*, 1st edn, Intrusion.Net Books, Sparta, N.J.

Anderson, J 1980, *Computer Security Threat Monitoring and Surveillance.*, Fort Washington.

Antonelli, C, Undy, M & Honeyman, P 1998, *The Packet Vault: Secure Storage of Network Data*, USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara.

Avolio, F 1999, 'Firewalls and Internet Security, the Second Hundred (Internet) Years', *The Internet Protocol Journal*, vol. 2, no. 2.

Axelsson, S 2000, *Intrusion Detection Systems: A Survey and Taxonomy*, Depart. of Computer Engineering, Chalmers University, G¨oteborg, Sweden.

Baldi, M, Gai, S & Picco, G 1997, 'Exploiting Code mobility in Decentralised and Flexible Network Management', paper presented to Workshop on Mobile Agents, Berlin, April.

Bass, T 2000, 'Intrusion detection systems and multisensor data fusion', *Communications of the ACM*, vol. 43, no. 4, pp. 99-105.

Bishop, M 1995, 'A Standard Audit Trail Format', paper presented to In Proceedings of the 18th National Information Systems Security Conference, October 10-13 1995.

Brox, A 2002, 'Signature Based or Anomaly Based Intrusion Detection – The Practice and Pitfalls', *Schmagazine*.

CERT® 2004, *CERT*, <http://www.cert.org/>.

Chapman, DB & Zwicky, ED 1995, *Building Internet firewalls*, Minor corrections, Nov. 1995. edn, O'Reilly & Associates, Cambridge.

Cheswick, B 1990, 'The Design of a Secure Internet Gateway', paper presented to 1990 USENIX Summer Conference.

Cheswick, WR, Bellovin, SM & Rubin, AD 2003, *Firewalls and Internet Security Second Edition*, Second edn, Professional Computing Series, Addison-Wesley, Boston.

CSRC 1999, *Common Criteria for Information Technology Security Evaluation*.

Denning, D 1987, 'An Intrusion-Detection Model', *IEEE Transactions on Software Engineering*, vol. SE 13, no. 2, pp. 222-32.

Ertoz, L, Eilertson, E, Lazarevic, A, Tan, P, Dokas, P, Srivastava, J & Kumar, V 2003, *Detection and Summarization of Novel Network Attacks Using Data Mining*.

Ertoz, L, Eilertson, E, Lazarevic, A, Tan, P, Srivastava, J, Kumar, V & Dokas, P 2004, 'The MINDS - Minnesota Intrusion Detection System', in *Next Generation Data Mining*.

Garfinkel, S & Spafford, G 1991, *Practical UNIX security*, O'Reilly & Associates, Sebastopol, CA.

Grabnar, M 2004, 'File :: Tail - Perl extension for reading from continously updated files', in http://www.enstimac.fr/Perl/perl5.6.1/site_perl/5.6.1/File/Tail.html.

Heberlein, L, Dias, G, Levitt, K, Mukherjee, B, Wood, J & Wolber, D 1990, 'A Network Security Monitor', *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, pp. 296-304.

Hoffman, D, Prabhakar, D & Strooper, P 2003, 'Testing iptables', paper presented to IBM Centre for Advanced Studies Conference, Toronto, Ontario, Canada.

Hofmeyr, S, Forrest, S & Somayaji, A 1998, 'Intrusion Detection using Sequences of System Calls', *Journal of Computer Security*, vol. 6, p. 151/80.

Holden, G 2003, *Guide to Network Defence and Countermeasures*, Course Technology, Thomson.

Kahani, M & Beadle, H 1997, 'Decentralised Approaches for Network Management', *Computer Communications Review*, vol. 27, no. 3.

Kemmerer, R & Vigna, G 2002, 'Intrustion Detection: A Brief History and Overview', *IEEE Computer*, vol. Special publication on Security and Privacy.

Krügel, C & Toth, T 2002, 'Flexible, Mobile Agent based Intrusion Detection for Dynamic Networks', *European Wireless*.

Kumar, S 1995, 'Classification and detection of computer intrusions.' PhD thesis, Purdue University.

Lee, W 2002, 'Applying data mining to intrusion detection: the quest for automation, efficiency, and credibility', *ACM SIGKDD Explorations Newsletter*, vol. 4, no. 2, pp. 35-42.

Lee, W, Stolfo, S & Mok, K 1999, 'Mining in a data-flow environment: experience in network intrusion detection', paper presented to Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining.

Leyden, J 2004, *MyDoom is the worst virus ever*, The Register, viewed 3 May 2004, <http://www.theregister.co.uk/2004/01/28/mydoom_is_the_worst_virus/>.

Lorimer, S 2003, 'A Real-Time IDS Monitoring Multiple Gateways', Honours thesis, University of Tasmania.

Manderson, K 2004, to J Scanlan.

McCallam, D, Whitson, J & Zavidniak, P 2002, 'Real Time Intrusion Detection - Applying Correlation and Fusion for Outside the Network Attack Forecasting and Insider Attack Detection', paper presented to RTO Information Systems Technology Panel (IST) Symposium, Estoril, Portugal.

Micro, T 2002, *The Real Cost of a Virus Outbreak*, Trend Micro, Cupertino CA.

MINDS, RT 2004, *MINDS - Minnesota Intrusion Detection System*, <http://www.cs.umn.edu/research/minds/MINDS.htm>.

Mounji, A, LeCharlier, B, Zampunieris, D & Habra, N 1995, *Distributed Audit Trail Analysis*, Institut d'Imformatique, Namur, Belgium.

Netfilter 2004, *iptable homepage*, http://www.netfilter.org/.

Northcutt, S, Cooper, M, Fearnow, M & Fredrick, K 2001, *Intrusion Signatures and Analysis*, First edn, New Riders, Indiana.

Pfleeger, CP & Pfleeger, SL 2003, *Security in computing*, 3rd Int edn, Prentice Hall PTR, Upper Saddle River, N.J.

Ptacek, T & Newsham, T 1998, *Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection*, Secure Networks.

Ranum, M 1994, 'Thinking About Firewalls', paper presented to In Proceedings of Second International Conference on Systems and Network Security and Management (SANS-II), April 1993.

---- 2001a, 'Experiences Benchmarking Intrusion Detection Systems', *NFR Security Technical Publications*.

---- 2001b, 'Coverage in Intrusion Detection Systems', *NFR Security Technical Publications*.

Ranum, M & Avolio, F 1994, 'A Toolkit and Methods for Internet Firewalls', paper presented to In Technical Summer Conference, Boston, June 1994.

Rowland, C 2003, *PortSentry HomePage*, http://sourceforge.net/projects/sentrytools/.

Scambray, J, McClure, S & Kurtz, G 2001, *Hacking Exposed: Network Security Secrets and Solutions*, Second edn, Osbourne, Berkeley.

Schneier, B 2000, *Secrets and lies : digital security in a networked world*, John Wiley, New York ; Chichester.

Singh, S 1999, *The code book : the science of secrecy from ancient Egypt to quantum cryptography*, Fourth Estate, London.

Siraj, A, Vaughn, R & Bridges, S 2004, 'Intrusion Sensor Data Fusion in an Intelligent Intrusion Detection System Architecture', paper presented to 37th Hawaii International Conference on System Sciences, Hawaii, January 5-8.

Siyan, K & Hare, C 1995, *Internet firewalls and network security*, New Riders Pub., Indianapolis, Ind.

SourceFire 2004a, *Beyond Detection*, SourceFire Inc., Columbia.

---- 2004b, *Snort 2.0: Detection Revistited*, SourceFire Inc., Columbia.

Stallings, W 2003, *Network security essentials : applications and standards*, 2nd edn, Pearson Education, Upper Saddle River, NJ.

Stoll, C 1991, *The cuckoo's egg : tracking a spy through the maze of computer espionage*, Pan Books, London.

Twycross, J 2004, *Immune Systems, Danger Theory and Intrusion Detection*, University of Nottingham.

Vicomsoft 2004, *Firewall White Paper*, Vicomsoft.

Vigna, G & Kemmerer, R 1999, 'NetSTAT: A Network-based Intrusion Detection Approach', *Journal of Computer Security*, vol. 7, no. 1, pp. 37-71.

Vigna, G, Valeur, F & Kemmerer, R 2003, 'Designing and Implementing a Family of Intrusion Detection Systems', *Foundations of Software Engineering*, pp. 88-97.

8. Appendices

Appendix A: iptable Rules Generated by System

The following appendix will aim to illustrate how the System responds to two different attacking source IP addresses. The first (IP1) is actually one of the attackers which was examined in Section 5.5.2, '*Attack Timing*,' and the second (IP2) is an example of the form in which a large percentage of the attacks against the network take: a fast scan lasting under 1 min. The table below lists the intervals been probes sent by the two source IP addresses. The first few cells of each column are shaded, illustrating the section of the attack which occurred before being detected as a threat to multiple gateways by the Analysis Module. It is also this section upon which the initial ban calculation is made.

#	IP1	IP2
1	---	---
2	3	5
3	85757	2
4	3	--
5	21527	1
6	3	---
7	21637	2
8	3	3
9	21564	---
10	3	4
11	21587	3
12	3	1
13	21619	----
14	4	5
15	21569	
16	3	
17	21591	
18	3	

The Two Example Attacks (Time between probes in seconds)

IP1

IP1 was detected as being a threat to multiple gateways after 3 probes and 23 hours 49 min and 20 seconds. Therefore the system banned the source IP upon each of the gateways with the following rule (where IP1 is the actual IP address):

```
iptables -I INPUT -s IP1 -j DROP
```

The rule, loosely translated, means that all inward bound packets from IP1 will be dropped, and that no message will be sent to alert the user to it being dropped. In Section 5.5.1 '*REJECT vs DROP*' the word DROP was replaced with REJECT, which resulted in a message being sent to the source IP to inform them of their packet being dropped at the gateway.

Using the timing information based on the first 3 packets sent by the source IP it was calculated that the average time interval was 7 hours 56 min and 44 seconds. It is this value which was multiplied by the number 11 (as outlined in Section 5.3) and generated the ban length. As the length generated was greater then 24 hours, the user was then banned for 24 hours. After 24 hours the probe count reached 11, and an activity check was carried out and found that the last probe by the user was sent 1 min and 44 seconds before the ban liftTime. As a result the ban was not lifted, and a new liftTime was set for 24 hours afterwards. As the attack lasted only another 18 hours, the ban was lifted after the second activity check.

IP2

The second attack we examined sent 5 probes against the network before targeting multiple gateways. These initial 5 probes lasted a total of 8 seconds, with probes 3 and 4 occurring within the same second. The resulting average time interval for the attack was 1.6 seconds. As the System was designed to have a minimum ban length of 100 seconds to aid in catering for attacks which do occur extremely fast, and may

result in a 0 second ban length, it is worth noting that the additional 100 seconds does play a role in this ban length. The calculated ban length, based on the interval multiplied by 11 results in a ban length of 17 seconds; with the added 100 it becomes 117. The attack lasted for 18 seconds after the ban was put in place, thus the additional 100 seconds allowed for the entire attack to fall within the ban length calculated.

The iptable rules generated for each gateway for IP2 attacker are identical to that of the rules generated for IP1, with the only difference being the different source IP. Each gateway upon which the source IP is to be blocked gets its own copy of the rule.

Appendix B: Regular Expressions

The vast majority of the audit log parsing done within the System is done by the regular expressions which were written by Sam Lorimer in 2003. The one exception is the regular expression listed below which was used to extract the IP address of those who had been banned and were continuing their attack. The data that was gathered was used to create Figure 5.2 on page 52.

Line 4	Aug 1 12:33:50 ns1 portsentry[7688]: attackalert: Host: as132231.bb132.soon-net.com.hk/203.180.13.231 is already blocked Ignoring

Target Audit Log Entry

```
/                   # Expresion Ends
^                   # beginning-of-string anchor
(\S+)               # assigned to $month
[^\d]+              # move to the number representing the date
(\S+)               # assigned to $date
\                   # literal space
(\S+)               # assigned to $time
\                   # literal space
\S+                 # skip over "ns1" text
[^\[]+\[            # move to after the '[' literal
([^\]]+)            # assigned to $ps_code
\]:\                # ']:' and literal space
([^:]+)             # assigned to $alert
:\\S+ [^\/]+\/      # move to front of IP
(\d+)               # assigned to $ip1
\.                  # literal full stop
(\d+)               # assigned to $ip2
\.                  # literal full stop
(\d+)               # assigned to $ip3
\.                  # literal full stop
(\d+)               # assigned to $ip4
\S+                 # move to end
/x;                 # Expresion ends
```

The Regular Expression used to extract the IP, timestamp and other information.

Appendix C: Admin Panel

The System that was produced in order to undertake the research within this thesis allowed for the real-time detection of malicious source IP addresses who were targeting multiple gateways upon a single network. Once detection had occurred the System was then able to respond in the defense of the network, blocking the source IP's access to the gateways upon the network, in preemption of the user's probable continued attack. The system in the form it was used for the majority of the research is not designed to be particularly user friendly or visually representative of the current state of the system due to the study being aimed at merely meeting the goals as outlined in Section 1.1. However, for the System to be used upon actual live data it would be of great use to be able to view and change several of the settings within the System, from adding a ban or new friendly Gateway, to lifting a ban which was mistakenly added. The Admin Panel provides this functionality, and has sections titled: Overview, Friendly, Banned, and Ban History.

Overview

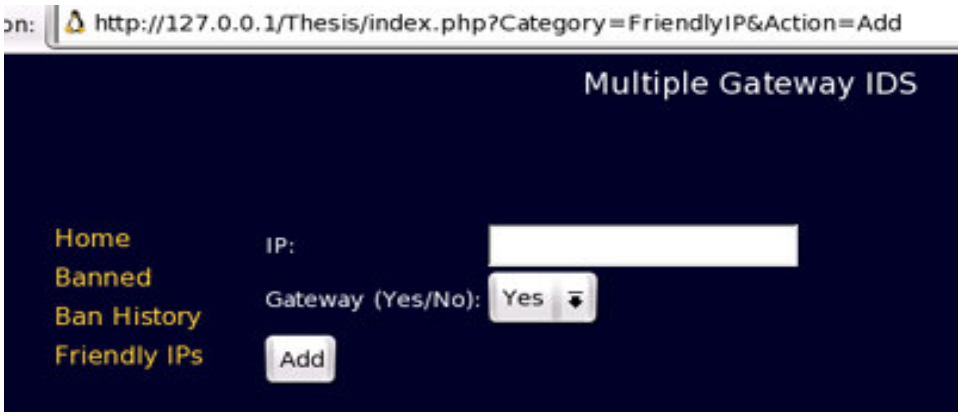
The front page of the Admin Panel displays the current status of the System, whether it is current running or not. Also on this front page it is possible to start or stop the System.

Friendly IPs

The Friendly IP section of the Admin Panel allows for the listing of Friendly IPs upon the network to be viewed. The Listing shows the IP of the friendly, the time they were added to the system, whether or not the given friendly is a gateway for rules to be set upon and a count of the total number of bans which have been set upon a given gateway.

Friendly IP Panel

Similar to the other sections within the Admin Panel, the Friendly IP section also allows for entries to be added. When adding a new source IP it gives the user a choice of dictating whether the given IP is a gateway or just a friendly IP address.



Add Friendly IP

Banned

The section of the Admin Panel displays all of the current bans which are in place upon the network. In addition to displaying the banned Source IP addresses it also shows their Attack Time, Lift Time, Set Time, Last Activity, Activity Count and the

rule which is in place for the IP. The listing (which shows 25 per page) can be sorted by any of these values (ascending or descending) to allow for easy browsing through the list by an Administrator.

Banned Panel

Above the IP listing and sort controls there is a link which takes the user to another page which allows for a ban to be added to the system, similar to the page for adding new Friendly IP addresses. The Banned screen also allows for any of these bans to be lifted at any time, by merely pressing the X in the appropriate source IP's table row.

Ban History

The Ban History Section is the visual representation of the history stored by the system of users who have been a threat to the system in the past. As it stores much of the same information as the Banned Panel it is not shown here. Similar to the other sections Ban History entries can be added or removed.